ITS
Institut
Teknologi
Sepuluh Nopember

TUGAS AKHIR - TE141599

PERANCANGAN DAN IMPLEMENTASI PENGATURAN KECEPATAN MOTOR *BRUSHLESS DC* MENGGUNAKAN METODE *MODEL PREDICTIVE CONTROL* (MPC)

Fachrul Arifin
NRP 2211100118

Dosen Pembimbing
Ir. Josaphat Pramudijanto, M.Eng.
Ir. Ali Fatoni, M.T.

JURUSAN TEKNIK ELEKTRO
Fakultas Teknologi Industri
Institut Teknologi Sepuluh Nopember
Surabaya 2015

ITS
Institut
Teknologi
Sepuluh Nopember

FINAL PROJECT - TE 141599

SPEED CONTROLLER DESIGN AND IMPLEMENTATION FOR BRUSHLESS DC MOTOR USING MODEL PREDICTIVE CONTROL (MPC)

Fachrul Arifin
NRP 2211100118

Advisors
Ir. Josaphat Pramudijanto, M.Eng.
Ir. Ali Fatoni, M.T.

DEPARTEMENT OF ELECTRICAL ENGINEERING
Faculty of Industrial Technology
Institut Teknologi Sepuluh Nopember
Surabaya 2015

PERANCANGAN DAN IMPLEMENTASI PENGATURAN KECEPATAN MOTOR *BRUSHLESS DC* MENGGUNAKAN METODE *MODEL PREDICTIVE CONTROL* (MPC)

TUGAS AKHIR

Diajukan Guna Memenuhi Sebagian Persyaratan Untuk
Memperoleh Gelar Sarjana Teknik Elektro
Pada
Bidang Studi Teknik Sistem Pengaturan
Jurusan Teknik Elektro
Institut Teknologi Sepuluh Nopember

Menyetujui

Dosen Pembimbing I,	Dosen Pembimbing II,
Ir. Josaphat Pramudijanto, M.Eng.	Ir. Ali Fatoni, M.T.
NIP. 196210051990031003	NIP. 196206031989031002

SURABAYA
JULI, 2015

PERANCANGAN DAN IMPLEMENTASI PENGATURAN KECEPATAN MOTOR *BRUSHLESS DC* MENGGUNAKAN METODE *MODEL PREDICTIVE CONTROL* (MPC)

Fachrul Arifin
2211100118

Dosen Pembimbing I : Ir. Josaphat Pramudijanto, M.Eng.
Dosen Pembimbing II : Ir. Ali Fatoni, M.T.

ABSTRAK

Saat ini, penelitian dan pengembangan mobil listrik (*electric vehicle*) sudah menjadi pusat perhatian bagi kalangan industri dan *civitas academica*. Sudah banyak kendaraan listrik yang sudah diproduksi secara massal di seluruh dunia. Beberapa dari kendaraan listrik tersebut menggunakan motor BLDC sebagai penggerak utamanya. Pada Tugas Akhir kali ini, salah satu kemampuan yang akan diteliti dan dianalisa adalah respon kecepatan motor BLDC pada mobil listrik saat mobil diberi suatu beban. Ketika mendapat pembebanan, respon kecepatan dari motor BLDC akan turun sehingga performa dari motor BLDC tidak sesuai dengan *setpoint* yang diharapkan.

Oleh karena itu, dibutuhkan suatu kontroler yang dapat mengatasi permasalahan diatas dan melakukan pengaturan kerja pada motor BLDC agar bekerja sesuai dengan kebutuhan. Kontroler ini diharapkan mampu untuk mengembalikan respon kecepatan pada motor BLDC kembali kepada setpoint ketika motor BLDC diberi pembebanan.

Penulis menggunakan metode kontroler *Model Predictive Control* (MPC) yang dapat memprediksi perilaku sistem pada masa depan yang bergantung pada informasi sistem saat ini dan model *state-space* dari sistem. Berdasarkan hasil implementasi, kontroler MPC dapat memberikan respon yang sesuai dengan *tracking setpoint* yang diberikan dengan rata-rata nilai *steady-state error* sebesar 9,8% untuk semua parameter pembebanan.

Kata Kunci : Kendaraan Listrik, *Brushless DC, Model Predictive Control*

PERANCANGAN DAN IMPLEMENTASI PENGATURAN KECEPATAN MOTOR *BRUSHLESS DC* MENGGUNAKAN METODE *MODEL PREDICTIVE CONTROL* (MPC)

Fachrul Arifin
2211100118

Dosen Pembimbing I : Ir. Josaphat Pramudijanto, M.Eng.
Dosen Pembimbing II : Ir. Ali Fatoni, M.T.

ABSTRAK

Saat ini, penelitian dan pengembangan mobil listrik (*electric vehicle*) sudah menjadi pusat perhatian bagi kalangan industri dan *civitas academica*. Sudah banyak kendaraan listrik yang sudah diproduksi secara massal di seluruh dunia. Beberapa dari kendaraan listrik tersebut menggunakan motor BLDC sebagai penggerak utamanya. Pada Tugas Akhir kali ini, salah satu kemampuan yang akan diteliti dan dianalisa adalah respon kecepatan motor BLDC pada mobil listrik saat mobil diberi suatu beban. Ketika mendapat pembebanan, respon kecepatan dari motor BLDC akan turun sehingga performa dari motor BLDC tidak sesuai dengan *setpoint* yang diharapkan.

Oleh karena itu, dibutuhkan suatu kontroler yang dapat mengatasi permasalahan diatas dan melakukan pengaturan kerja pada motor BLDC agar bekerja sesuai dengan kebutuhan. Kontroler ini diharapkan mampu untuk mengembalikan respon kecepatan pada motor BLDC kembali kepada setpoint ketika motor BLDC diberi pembebanan.

Penulis menggunakan metode kontroler *Model Predictive Control* (MPC) yang dapat memprediksi perilaku sistem pada masa depan yang bergantung pada informasi sistem saat ini dan model *state-space* dari sistem. Berdasarkan hasil implementasi, kontroler MPC dapat memberikan respon yang sesuai dengan *tracking setpoint* yang diberikan dengan rata-rata nilai *steady-state error* sebesar 9,8% untuk semua parameter pembebanan.

Kata Kunci : Kendaraan Listrik, *Brushless DC*, *Model Predictive Control*

SPEED CONTROLLER DESIGN AND IMPLEMENTATION FOR BRUSHLESS DC MOTOR USING MODEL PREDICTIVE CONTROL (MPC)

Fachrul Arifin
2211100118

Advisor I : Ir. Josaphat Pramudijanto, M.Eng.
Advisor II : Ir. Ali Fatoni, M.T.

ABSTRACT

Nowadays, the research and development about electric vehicle becoming an interesting topic for researcher, students and global industry around the world. Most of the electric vehicle uses BLDC motor tchnology as their main powerdrive in the electric vehicle. As a future technology, electric vehicle have a high expectation as new transportation techonology to answer a global challenge and provide a better and greener transportation techonlogy around the world.

This Final Project will discuss about the BLDC Motor ability and speed response on load condition. On load condition, the speed response from BLDC motor will decrease and doesn't meet the criteria that we expected from. Because of that, the BLDC motor need some controller to eliminate the difference between the actual response and desired setpoint.

In this final project, the writer designed some controller based on Model Predictive Control (MPC) method to eliminate those error. The MPC method have some formula to predict the future behaviour of the plant output relies on the state-space model and current plant information. After some simulation and implementation, the MPC controller is capable to bringing back the actual response (in this case the speed of BLDC motor) to the desired setpoint with average 9.8% steady-state error in all load condition.

Keywords **:** *Electric Vehicle, Brushless DC, Model Predictive Control*

KATA PENGANTAR

Alhamdullilah, puji syukur penulis panjatkan kehadirat Allah SWT karena atas rahmat dan karunia-Nya penulis dapat menyelesaikan penulisan buku Tugas Akhir dengan judul **"PERANCANGAN DAN IMPLEMENTASI PENGATURAN KECEPATAN MOTOR *BRUSHLESS DC* MENGGUNAKAN METODE *MODEL PREDICTIVE CONTROL* (MPC)"**. Tugas akhir merupakan salah satu syarat yang harus dipenuhi untuk menyelesaikan program studi Strata-1 pada Jurusan Teknik Elektro Fakultas Teknologi Industri Institut Teknologi Sepuluh Nopember Surabaya.

Penulis menyadari bahwa dalam penulisan skripsi ini banyak mengalami kendala, namun berkat bantuan, bimbingan, dan kerjasama dari berbagai pihak sehingga kendala-kendala tersebut dapat diatasi. Untuk itu pada kesempatan ini penulis menyampaikan banyak terimakasih dan penghargaan setingi-tingginya kepada :

1. Kedua orang tua, Ayahanda Piping Zaenal Aripin dan Ibunda Sofiarini serta adikku tercinta Lina Maghfira yang selalu memberikan dukungan, semangat, dan doa kepada penulis.
2. Bapak Josaphat Pramudijanto dan Bapak Ali Fatoni selaku Dosen Pembimbing atas segala bantuan, perhatian, dan arahan selama pengerjaan Tugas Akhir ini.
3. Bapak Rusdhianto Effendie selaku Koordinator Bidang Studi Teknik Sistem Pengaturan Jurusan Teknik Elektro ITS dan Bapak Tri Arief Sardjono selaku Ketua Jurusan Teknik Elektro ITS.
4. Bapak dan Ibu dosen bidang studi Teknik Sistem Pengaturan, Teknik Elektro ITS.
5. Beny, Habib, Huda dan Ammar sebagai BLDC Team yang sangat solid dalam mendesain dan mewujudkan *plant* BLDC.
6. Mas Fahrul Abbas sebagai pembimbing S2 yang sangat membantu dalam mewujudkan berdirinya *plant* BLDC ini.
7. Serta rekan-rekan yang tidak saya bisa sebutkan satu-persatu.

Penulis menyadari bahwa pada penyusunan laporan tugas akhir ini masih terdapat kekurangan-kekurangan karena keterbatasan kemampuan yang penulis miliki, walaupun demikian penulis berharap tugas akhir ini dapat bermanfaat bagi yang membutuhkannya.

Surabaya, Juli 2015
Penulis

DAFTAR ISI

HALAMAN

HALAMAN JUDUL ..... **i**
PERNYATAAN KEASLIAN TUGAS AKHIR ..... **v**
LEMBAR PENGESAHAN ..... **vii**
ABSTRAK ..... **ix**
ABSTRACT ..... **xi**
KATA PENGANTAR ..... **xiii**
DAFTAR ISI ..... **xv**
DAFTAR GAMBAR ..... **xix**
DAFTAR TABEL ..... **xxi**

BAB 1. PENDAHULUAN **1**
1.1 Latar Belakang 1
1.2 Perumusan Masalah 2
1.3 Batasan Masalah 2
1.4 Tujuan Penelitian 2
1.5 Sistematika Penulisan 3
1.6 Relevansi 4

BAB 2. TEORI PENUNJANG **5**
2.1 Motor *Brushless* DC 5
2.1.1 Konstruksi Motor *Brushless* DC 5
2.1.2 *Driver* dan Kontroler Motor *Brushless* DC 8
2.1.3 Prinsip Kerja Motor *Brushless* DC 9
2.2 Rem Elektromagnetik 10
2.3 Sensor *Rotary Encoder* 11
2.4 Arduino Uno 12
2.5 MATLAB R2014a 13
2.6 Identifikasi Sistem 14
2.6.1 Identifikasi Dinamis 15
2.6.1.1 Sinyal *Pseudo-Random Binary Sequence* 16
2.6.1.2 *Auto-Regressive Exogeneous* (ARX) 16
2.6.2 Validasi Model 17

2.7 *Model Predictive Control* (MPC) 18
2.7.1 Model *State-Space* dengan *Embedded Integrator* 19
2.7.2 Perhitungan *Prediction Output* & *Future Control* 20
2.7.3 Indeks Performansi Kontroler MPC 21
2.7.4 *Closed-loop Control System* 22
2.7.5 *State Estimation* 24

BAB 3. PERANCANGAN SISTEM 27
3.1 Gambaran Umum Sistem 27
3.2 Perancangan Perangkat Keras 28
3.2.1 Perancangan Mekanik 28
3.2.1.1 Motor *Brushless DC* 28
3.2.1.2 *Electronic Speed Control* (ESC) 29
3.2.1.3 Rem Elektromagnetik 31
3.2.2 Perancangan Elektronik 32
3.2.2.1 Rangkaian Sensor *Rotary Encoder* 32
3.2.2.2 Rangkaian Penyearah Gelombang AC 33
3.2.2.3 Rangkaian *Driver* Rem Elektromagnetik 34
3.2.2.4 Rangkaian Sensor Arus 35
3.2.2.5 Mikrokontroler Arduino 36
3.3 Perancangan Perangkat Lunak 37
3.3.1 *Software* Arduino 37
3.3.2 *Software* MATLAB 37
3.4 Identifikasi dan Pemodelan Sistem 39
3.4.1 Metode Pembebanan *Plant* 39
3.4.2 Metode Identifikasi dan Pemodelan 39
3.4.3 Pemodelan Motor *Brushless* DC 40
3.4.3.1 Beban Minimal 40
3.4.3.2 Beban Nominal 41
3.4.3.3 Beban Maksimal 42
3.4.4 Pengujian dan Validasi 42
3.5 Perancangan Kontroler *Model Predictive Control* 43
3.5.1 Perancangan Model *State Space* 43
3.5.2 Desain *Augmented Model* 44
3.5.3 Penentuan Parameter dan *Gain* Kontroler MPC 45
3.5.4 Desain *State Estimation* Menggunakan *Observer* 47

BAB 4. PENGUJIAN DAN ANALISA 49
4.1 Gambaran Umum Pengujian Sistem 49
4.2 Pengujian Perangkat Keras 50
4.2.1 Pengujian *Electronic Speed Control* Motor BLDC 50
4.3 Simulasi Sistem 51
4.3.1 Diagram Blok Simulasi Sistem 51
4.3.2 Prosedur Pengerjaan Simulasi Sistem 53
4.4 Pengaruh Parameter N_p pada Sistem 53
4.4.1 Beban Minimal 55
4.4.2 Beban Nominal 55
4.4.3 Beban Maksimal 56
4.5 Pengaruh Parameter N_c pada Sistem 57
4.5.1 Beban Minimal 57
4.5.2 Beban Nominal 59
4.5.3 Beban Maksimal 60
4.6 Pengaruh Parameter r_w pada Sistem 60
4.6.1 Beban Minimal 62
4.6.2 Beban Nominal 62
4.6.3 Beban Maksimal 63
4.7 Respon *Tracking* Sistem 64
4.7.1 Beban Minimal 64
4.7.2 Beban Nominal 65
4.7.3 Beban Maksimal 66
4.8 Implementasi Sistem 67
4.8.1 Realisasi Perancangan Sistem 67
4.8.2 Diagram Blok Implementasi Sistem 71
4.8.3 Analisa Kontroler MPC pada Implementasi Sistem 73
4.8.3.1 Beban Minimal 74
4.8.3.2 Beban Nominal 75
4.8.3.3 Beban Maksimal 77

BAB 5. PENUTUP 79
5.1 Kesimpulan 79
5.2 Saran 80

DAFTAR PUSTAKA 81
LAMPIRAN 83
RIWAYAT HIDUP 89

DAFTAR TABEL

Tabel 3.1 Spesifikasi Motor BLDC RCTimer HP2212-1000KV..... 29
Tabel 3.2 Spesifikasi ESC Turnigy PLUSH-25A. 30
Tabel 3.3 Spesifikasi Sensor Arus SparkFun ACS712 5A. 36
Tabel 3.4 Metode Pembebanan pada Sistem. 39
Tabel 3.5 Identifikasi Dinamis pada Beban Minimal. 41
Tabel 3.6 Identifikasi Dinamis pada Beban Nominal. 42
Tabel 3.7 Identifikasi Dinamis pada Beban Maksimal. 42
Tabel 3.8 Pemodelan Motor BLDC pada Berbagai Macam Kondisi Pembebanan 43
Tabel 4.1 Hubungan Nilai *Input* PWM, Tegangan *Output* ESC dan Kecepatan Motor BLDC. 51
Tabel 4.2 Karakteristik Respon Sistem pada Pembebanan Minimal untuk Tiap-Tiap Variabel *Prediction Horizon*. 55
Tabel 4.3 Karakteristik Respon Sistem pada Pembebanan Nominal untuk Tiap-Tiap Variabel *Prediction Horizon*. 56
Tabel 4.4 Karakteristik Respon Sistem pada Pembebanan Maksimal untuk Tiap-Tiap Variabel *Prediction Horizon*. 56
Tabel 4.5 Karakteristik Respon Sistem pada Pembebanan Minimal untuk Tiap-Tiap Variabel *Control Horizon*. 59
Tabel 4.6 Karakteristik Respon Sistem pada Pembebanan Nominal untuk Tiap-Tiap Variabel *Control Horizon*. 59
Tabel 4.7 Karakteristik Respon Sistem pada Pembebanan Maksimal untuk Tiap-Tiap Variabel *Control Horizon*. 60
Tabel 4.8 Karakteristik Respon Sistem pada Pembebanan Minimal untuk Tiap-Tiap Variabel *Tuning Parameter*. 62
Tabel 4.9 Karakteristik Respon Sistem pada Pembebanan Nominal untuk Tiap-Tiap Variabel *Tuning Parameter*. 63
Tabel 4.10 Karakteristik Respon Sistem pada Pembebanan Maksimal untuk Tiap-Tiap Variabel *Tuning Parameter*. 63
Tabel 4.11 Karakteristik Respon Sistem pada Pembebanan Minimal dengan Referensi *Tracking*. 64
Tabel 4.12 Karakteristik Respon Sistem pada Pembebanan Nominal dengan Referensi *Tracking*. 65
Tabel 4.13 Karakteristik Respon Sistem pada Pembebanan Maksimal dengan Referensi *Tracking*. 66

Tabel 4.14 Nilai Parameter dari Kontroler MPC untuk Tiap–Tiap Nilai Pembebanan. 73
Tabel 4.15 Karakteristik Respon dengan Pembebanan Minimal pada Implementasi Sistem. 74
Tabel 4.16 Karakteristik Respon dengan Pembebanan Nominal pada Implementasi Sistem. 76
Tabel 4.17 Karakteristik Respon dengan Pembebanan Maksimal pada Implementasi Sistem 78

DAFTAR GAMBAR

Gambar 2.1 Konstruksi Umum Motor *Brushless* DC (BLDC)..... 6
Gambar 2.2 Konstruksi *Brushless* DC (BLDC) Tipe *Outer Rotor*.... 6
Gambar 2.3 *Timing* Komutasi *Stator* pada Motor BLDC..... 7
Gambar 2.4 Skema Rangkaian *Driver* & Kontroler Motor BLDC.... 8
Gambar 2.5 Prinsip Kerja Motor BLDC..... 10
Gambar 2.6 Struktur dari Sebuah Rem Elektromagnetik. 11
Gambar 2.7 Konstruksi dan Prinsip Kerja Sensor *Rotary Encoder*. 12
Gambar 2.8 Dialog Tampilan *Instrument Control Toolbox*..... 14
Gambar 2.9 Sinyal Uji *Pseudo-Random Binary Sequence* (PRBS). 17
Gambar 2.10 Konsep dari Kontroler *Model Predictive Control*..... 18
Gambar 2.11 Diagram Blok Sistem Kontrol *Loop* Tertutup *Model Predictive Control* untuk Waktu Diskrit..... 23
Gambar 2.12 Diagram Blok Sistem Kontrol *Loop* Tertutup *Model Predictive Control* untuk Waktu Diskrit dengan menggunakan *Observer*..... 25
Gambar 3.1 Blok Diagram Sistem Pengaturan Kecepatan Motor *Brushless DC* (BLDC)..... 27
Gambar 3.2 Motor *Brushless* DC Tipe *Outrunner* dengan Tipe RCTimer HP2212-1000KV..... 29
Gambar 3.3 *Electronic Speed Control* Turnigy PLUSH-25A. 30
Gambar 3.4 Perancangan Konstruksi Fisik Rem Elektromagnetik.. 31
Gambar 3.5 Sensor *Rotary Encoder* yang Terpasang pada *Shaft*.... 32
Gambar 3.6 Skema Rangkaian *Rotary Encoder* yang Tersusun dari *Optocoupler* dan Resistor. 33
Gambar 3.7 Rangkaian Penyearah Gelombang AC yang Terdiri dari *Diode Bridge* dan Kapasitor. 33
Gambar 3.8 Skema Rangkaian *Driver* Rem Elektromagnetik..... 34
Gambar 3.9 Sensor Arus SparkFun ACS712 5A Breakout 35
Gambar 3.10 Mikrokontroler Arduino Uno R3..... 36
Gambar 3.11 Tampilan Arduino IDE 38
Gambar 3.12 Blok Identifikasi Sistem *Open Loop* pada Simulink.... 38
Gambar 3.13 Hubungan *Input-Output* Sistem pada Identifikasi Dinamis..... 40
Gambar 3.14 Perbandingan Respon Hasil Pengukuran dan Pemodelan pada Beban Minimal..... 41

Gambar 4.1 Realisasi *Plant* atau Sistem Pengaturan Kecepatan Motor *Brushless DC.* 49
Gambar 4.2 Diagram Blok Simulasi Pengujian Kontroler MPC pada Sistem..... 52
Gambar 4.3 Subsistem *Observer* pada Simulasi Sistem 52
Gambar 4.4 Pengaruh Parameter *Prediction Horizon* pada Sistem.. 54
Gambar 4.5 Sinyal Kontrol pada Sistem dengan Perubahan Parameter *Prediction Horizon*. 54
Gambar 4.6 Pengaruh *Control Horizon* pada Sistem. 58
Gambar 4.7 Sinyal Kontrol pada Sistem dengan Perubahan Parameter *Control Horizon*. 58
Gambar 4.8 Pengaruh Parameter *Tuning* Indeks Performansi pada Sistem..... 61
Gambar 4.9 Sinyal Kontrol pada Sistem dengan Perubahan *Tuning* Parameter pada Indeks Performansi. 61
Gambar 4.10 Respon Sistem pada Pembebanan Minimal dengan Referensi *Tracking*. 65
Gambar 4.11 Respon Sistem pada Pembebanan Nominal dengan Referensi *Tracking*. 66
Gambar 4.12 Respon Sistem pada Pembebanan Maksimal dengan Referensi *Tracking*. 67
Gambar 4.13 Realisasi Perancangan Sistem Pengaturan Kecepatan Motor BLDC. 68
Gambar 4.14 Konstruksi Rem Elektromagnetik pada Sistem. 69
Gambar 4.15 Hasil Perancangan Sensor *Rotary Encoder*. 69
Gambar 4.16 Alat Akuisisi Data berupa Mikrokontroler Arduino..... 70
Gambar 4.17 Rangkaian *Rectifier* pada Sistem..... 71
Gambar 4.18 Rangkaian *Driver* untuk Pengaturan Tegangan *Input* pada Rem Elektromagnetik. 71
Gambar 4.19 Diagram Blok Implementasi Kontroler MPC pada Sistem..... 72
Gambar 4.20 Subsistem *Plant* yang Terdiri Dari Komunikasi *Serial* pada Sistem. 72
Gambar 4.21 Hasil Implementasi Kontroler MPC pada Sistem dengan Pembebanan Minimal. 75
Gambar 4.22 Hasil Implementasi Kontroler MPC pada Sistem dengan Pembebanan Nominal..... 76
Gambar 4.23 Hasil Implementasi Kontroler MPC pada Sistem dengan Pembebanan Maksimal..... 77

RIWAYAT HIDUP

Fachrul Arifin, lahir di Cimahi, 10 April 1993. Riwayat pendidikannya, menamatkan pendidikan dasar di SD Negeri 1 Rancabelut Cimahi (tahun 2005), pendidikan menengah di SMP Negeri 1 Cimahi (tahun 2008), dan pendidikan tinggi di SMA Negeri 2 Bandung (tahun 2011). Saat ini telah menempuh kuliah di Jurusan Teknik Elektro Institut Teknologi Sepuluh Nopember dengan mengabil bidang studi Teknik Sistem Pengaturan. Selama kuliah penulis aktif dalam beberapa kegiatan akademis maupun non akademis. Penulis aktif dalam Himpunan Mahasiswa Teknik Elektro bidang Lingkar Kampus dan pernah menjabat sebagai Sekreteris Departemen periode 2013-2014. Penulis juga aktif dalam kegiatan keilmiahan seperti seminar dan berbagai macam pelatihan bertema keilmiahan. Penulis dapat dihubungi melalui email: **fachrul.arifin@live.com**

BAB 1
PENDAHULUAN

Bab 1 pada laporan Tugas Akhir ini akan menjelaskan latar belakang yang akan diangkat menjadi suatu rumusan masalah pada Tugas Akhir kali ini. Setelah rumusan masalah terbentuk, Bab 1 juga akan menjelaskan mengenai tujuan dan batasan masalah yang akan dikerjakan pada Tugas Akhir kali ini. Terakhir, pada Bab 1 juga dijelaskan mengenai sistematika penulisan dan relevansi pengerjaan Tugas Akhir ini.

1.1 Latar Belakang

Saat ini, seperti kita telah ketahui bersama bahwa cadangan dan persediaan bahan bakar fosil dari tahun ke tahun cenderung menipis. Kabar ini tentu saja menjadi hal yang buruk bagi dunia otomotif yang mengandalkan konsumsi bahan bakar fosil yang saat ini menjadi sumber utama penggerak kendaraan bermotor. Tentu saja, hal ini mendorong umat manusia untuk mencari bahan bakar alternatif dan lebih efisien dibandingkan bahan bakar fosil. Saat ini, telah banyak ditemukan berbagai macam kendaraan bermotor berbasiskan energi alternatif yang lebih murah dan efisien dibandingkan bahan bakar fosil. contohnya adalah mobil listrik yang saat ini menjadi tren dan diproyeksikan menjadi kendaraan masa depan.

Kendaraan listrik merupakan kendaraan yang menggunakan murni energi listrik sebagai penggeraknya. Pada kendaraan konvensional, bahan bakar dari minyak bumi yang telah diproses, contohnya premium dan pertama, digunakan untuk menjalankan mesin diesel biasa. Sedangkan pada kendaraan listrik, digunakan sumber energi listrik yang berasaldari baterai untuk menggerakkan kendaraan listrik tersebut.

Salah satu penggerak utama dari kendaraan listrik adalah Motor *Brushless* DC (BLDC) yang berfungsi sebagai komponen utama penggerak mobil listrik ini. Motor BLDC merupakan pengembangan dari motor DC konvensional atau *brushed* DC motor. Pada mobil konvensional berbahan bakar fosil, motor BLDC banyak digunakan dalam komponen penyusun mobil, seperti *Air Conditioner* (AC), *airbag*, *wiper blades*, dan macam lainnya. Kedepannya, motor BLDC diharapkan dapat menjadi penggerak utama pada mobil listrik walaupun kondisi jalan yang menantang sekalipun [1].

Pada rencana tugas akhir kali ini, akan diterapkan sebuah kontroler berbasiskan *Model Predictive Control* (MPC) untuk dapat menggerakkan motor BLDC secara optimal pada kondisi berbeban. MPC dipilih karena konsepnya yang sangat intuitif dan penalaannya yang mudah. Selain itu, MPC juga dapat menangani sistem dengan memperhitungkan batasan atau *constraint* [1]. Harapan dari implementasi kontroler MPC pada motor BLDC adalah didapatkannya respon motor yang cepat dan sesuai dengan kebutuhan mobil listrik.

1.2 Perumusan Masalah

Apabila suatu motor BLDC tidak diberikan beban, tentu saja kecepatan motor akan berjalan konstan dan *transfer function* dari motor BLDC akan berjalan dengan semestinya. Akan tetapi, di saat motor BLDC diberikan suatu beban, maka kecepatan motor akan berubah dan membuat *transfer function* motor bernilai tidak linear. Efek pembebanan ini dapat terjadi dikarenakan kondisi beban yang berubah-ubah pada mobil, contohnya akibat penambahan beban pada konstruksi mobil ataupun beban penumpang yang semakin bertambah. Tentu saja efek ini tidak diharapkan dikarenakan dapat mengganggu kinerja dan performa berupa penurunan kecepatan dari motor BLDC. Kontroler *Model Predictive Control* ini diharapkan dapat mengurangi dan mengeliminasi *error* yang timbul akibat perubahan beban yang ditanggung Motor BLDC.

1.3 Batasan Masalah

Berdasarkan rumusan masalah, penulis akan membatasi permasalahan yang akan diteliti sehingga tujuan dari penelitian dapat dicapai. Batasan dari penelitian tersebut adalah sebagai berikut:

a. *Range* kecepatan dari motor BLDC yang akan diteliti berada pada *range* kecepatan 1500-2500 RPM.
b. Metode yang akan digunakan pada sistem pengaturan kecepatan motor BLDC adalah *Model Predictive Control* menggunakan mikrokontroler Arduino Uno dan *software* MATLAB.
c. Penambahan beban yang akan digunakan pada motor BLDC menggunakan rem elektromagnetik dengan 3 variabel beban, yaitu minimal, nominal dan maksimal.

1.4 Tujuan Penelitian

Tujuan dari pelaksanaan Tugas Akhir ini adalah merancang dan mengimplementasikan suatu sistem pengaturan kecepatan motor BLDC

dengan menggunakan metode *Model Predictive Control* untuk menghilangkan *error* penurunan kecepatan akibat adanya beban.

1.5 Sistematika Penulisan

Sistematika penulisan pada laporan Tugas Akhir ini terdiri atas 5 bab, seperti yang dapat di lihat pada uraian berikut ini :

BAB 1 : PENDAHULUAN

Pada bab ini, akan dijelaskan mengenai latar belakang serta perumusan dan batasan masalah pada Tugas Akhir ini. Selain itu, akan dijabarkan pula tujuan dari Tugas Akhir ini beserta metodologi yang digunakan. Terakhir, akan dijelaskan pula mengenai sistematika penulisan dan relevansi Tugas Akhir ini.

BAB 2 : TEORI PENUNJANG

Bab ini menjelaskan tentang landasan teori yang diadopsi dan dipelajari pada pelaksanaan Tugas Akhir ini. Materi yang akan dijelaskan antara lain teori mengenai motor BLDC beserta perangkat keras pendukungnya serta metode dan perangkat lunak yang digunakan untuk pengaturan kecepatan motor BLDC.

BAB 3 : PERANCANGAN SISTEM

Bab ini menjelaskan tentang perancangan keras dan perangkat lunak yang akan digunakan pada pelaksanaan Tugas Akhir. Selain itu, akan dijabarkan pula mengenai perancangan kontroler menggunakan metode *Model Predictive Control* (MPC).

BAB 4 : PENGUJIAN DAN ANALISA

Pada bab ini, akan dijabarkan mengenai hasil simulasi kontroler pada *software* beserta analisanya. Selain itu, bab ini juga berisi tentang hasil implementasi kontroler pada *plant* motor BLDC beserta analisanya.

BAB 5 : PENUTUP

Bab terakhir ini akan menjelaskan tentang penarikan kesimpulan pelaksanaan Tugas Akhir dan saran untuk penelitian selanjutnya.

1.6 Relevansi

Hasil yang diperoleh dari pelaksanaan Tugas Akhir ini diharapkan menjadi referensi perancangan kontroler untuk mengatur kecepatan dari motor BLDC pada skala kendaraan listrik sesungguhnya. Selain itu, hasil dari penelitian ini juga diharapkan menjadi referensi perancangan kontroler MPC untuk pengaturan kecepatan jenis motor lainnya.

BAB 2
TEORI PENUNJANG

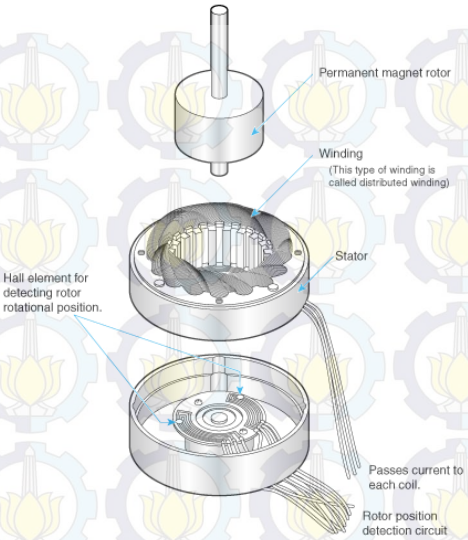
Bab 2 pada laporan Tugas Akhir ini akan menjelaskan mengenai topik dan teori umum yang menyangkut pelaksanaan Tugas Akhir kali ini. Pada bagian awal akan dijelaskan mengenai konstruksi dan cara kerja motor BLDC. Setelah itu, dideskripsikan pula mengenai identifikasi dan pemodelan suatu sistem menggunakan identifikasi dinamis. Terakhir, akan dijelaskan mengenai struktur dari kontroler yang berbasis *Model Predictive Control* atau MPC.

2.1 Motor *Brushless* DC

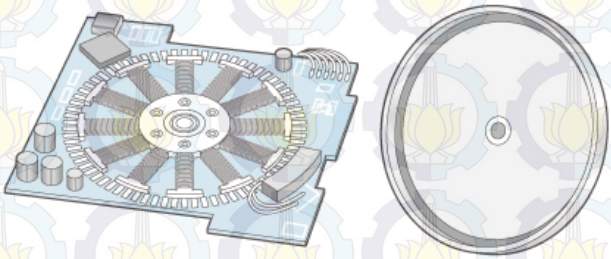
Motor *Brushless* DC (BLDC) merupakan salah satu jenis motor DC yang konstruksinya menggunakan magnet permanen di bagian *rotor* dan Kumparan jangkar pada *stator*. Motor BLDC tidak menggunakan fungsi *brush* sebagai media eksitasi ke *rotor*-nya, namun fungsi tersebut digantikan oleh medan magnet yang telah ditimbulkan oleh magnet permanen pada bagian *rotor*-nya. Keuntungan yang paling jelas dari konfigurasi *brushless* adalah penghapusan *brush*, yang memudahkan perawatan dan menghilangkan rugi gesek akibat adanya kontak antara *rotor* dan *brush*. Motor BLDC lebih baik dibandingkan dengan motor induksi, motor BLDC memiliki efisiensi yang lebih baik dan faktor daya yang lebih baik dan, oleh karena itu, daya *output* yang lebih besar untuk *frame* yang sama. [2]

2.1.1 Konstruksi Motor *Brushless* DC

Seperti yang telah dijelaskan pada paragraf sebelumnya, perbedaan mendasar antara motor DC konvensional dan motor BLDC adalah pada media eksitasinya. Untuk menggantikan peran *brush* seperti yang terdapat pada motor DC konvensional, digunakanlah rangkaian penggerak (*driver*) yang terdiri dari komponen semikonduktor atau MOSFET untuk menjalankan pensaklaran arus pada kumparan motor BLDC. Tapi rangkaian *driver* tersebut, motor BLDC tidak akan dapat berjalan sebagaimana mestinya. Secara garis besar, konstruksi umum motor BLDC dapat dilihat pada Gambar 2.1 dan Gambar 2.2. Terlihat pada gambar, motor BLDC terdiri dari 3 bagian utama, yaitu *rotor* yang berupa magnet permanen, kumparan jangkar atau *stator*, dan sensor posisi *rotor* yang berupa berupa sensor *hall-effect*.



Gambar 2.1 Konstruksi Umum Motor *Brushless* DC (BLDC). [3]



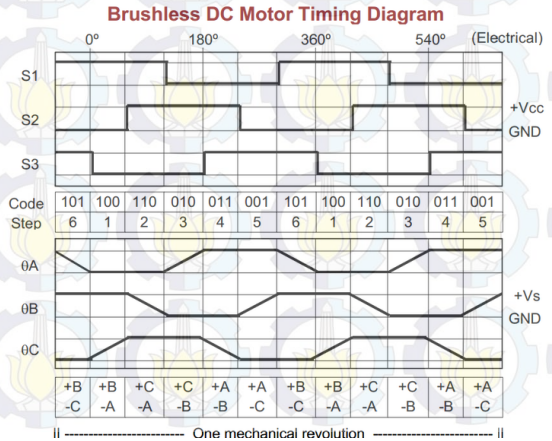
Gambar 2.2 Konstruksi *Brushless* DC (BLDC) Tipe *Outer Rotor*. [3]

Pada Tugas Akhir kali ini, akan dijelaskan mengenai konstruksi motor BLDC yang digunakan, yaitu motor BLDC tipe *radial flux* dan *outer rotor*. Motor yang digunakan pada Tugas Akhir ini dirancang untuk mengalirkan fluks secara *radial*. Dan konstruksi motor BLDC yang digunakan adalah tipe *outer rotor*. Tipe *outer rotor* memiliki kumparan

jangkar yang diletakkan pada bagian dalam motor. Sedangkan bagian *rotor* diletakkan pada bagian luar motor. Konstruksi motor BLDC tipe *outer rotor* dapat dilihat pada Gambar 2.2.

Stator pada motor BLDC merupakan rangkaian yang terdiri dari beberapa belitan yang disebut kumparan jangkar. Jumlah pasang kutub yang diciptakan di *stator* akibat dari arah dan jumlah lilitan berpengaruh terhadap performa dari motor BLDC. *Stator* pada motor BLDC terdiri dari 3 fasa yang biasanya terhubung dengan tiga buah kabel untuk disambungkan pada rangkaian kontrolnya. *Rotor* pada motor BLDC terdiri dari beberapa magnet permanen. Jumlah kutub magnet di *rotor* juga mempengaruhi ukuran langkah dan torsi dari motor. Magnet permanen pada motor BLDC biasanya menggunakan jenis magnet neodymium yang merupakan jenis magnet tetap yang sangat kuat.

Untuk menentukan orientasi posisi *rotor* fungsi komutasi dilakukan oleh sensor, terdapat beberapa jenis sensor yang digunakan. Misalkan sensor *optical encoder, magnetic encoder* atau *hall effect* yang berfungsi memberikan sinyal digital akibat adanya medan magnetik yang tegak lurus terhadap sensor. Sensor *hall* ini harus diletakkan sedekat mungkin dengan *rotor* magnet permanen untuk mendeteksi posisi kutub magnet pada *rotor*. Output *hall* akan akan dibaca oleh *decoder* yang nanti akan menentukan *timing* komutasi seperti pada Gambar 2.3 dan menentukan kumparan *stator* mana yang akan dinyalakan sehingga motor BLDC dapat berputar.

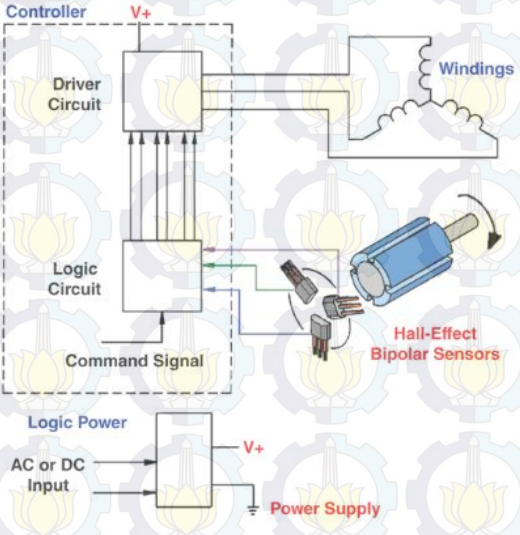


Gambar 2.3 *Timing* Komutasi *Stator* pada Motor BLDC. [4]

2.1.2 *Driver* dan Kontroler Motor *Brushless* DC

Driver atau kontroler pada motor BLDC sangatlah penting karena rangkaian inilah yang menggantikan peran *brush* pada motor DC konvensional untuk menyegerakan stator. Motor BLDC memerlukan suatu *trigger* pulsa yang masuk ke bagian stator motor BLDC untuk memberikan pengaturan besarnya arus yang mengalir sehingga motor dapat diatur secara akurat. Selain itu, *driver* pada motor BLDC juga berperan untuk mengubah tegangan DC menjadi tegangan AC dikarenakan motor BLDC memiliki kumparan stator 3 fasa pada konstruksinya. Secara umum, skema rangkaian kontrol dan *driver* pada motor BLDC dapat dilihat pada Gambar 2.4.

Pada Gambar 2.4, terlihat bahwa kontroler motor BLDC terdiri dari 2 bagian, yaitu rangkaian *driver* (*driver circuit*) dan rangkaian logika (*logic circuit*). Rangkaian *driver* biasanya terdiri dari beberapa transistor MOSFET (*metal–oxide–semiconductor field-effect transistor*) atau IGBT (*insulated-gate bipolar transistor*) yang berfungsi untuk merubah arus DC yang diberikan rangkaian logika dan mengubahnya menjadi arus AC dengan perbedaan fasa 120^{o} untuk menjalankan motor BLDC.



Gambar 2.4 Skema Rangkaian *Driver* & Kontroler Motor BLDC. [5]

Pada kontroler motor BLDC, rangkaian logika mempunyai peranan yang sangat penting untuk mengatur *timing* komutasi dan waktu *switching* yang tepat pada tiap fasa *stator* motor BLDC. Rangkaian logika ini bisa berupa mikroprosesor atau mikrokontroler yang menerima *input* dari sensor *hall effect* pada motor atau tegangan *back-emf* pada motor. *Input* ini akan diolah sedemikian rupa sehingga rangkaian logika ini dapat memberi *output* perintah kepada rangkaian *driver* untuk menyalakan kumparan stator tertentu pada motor BLDC. Saat ini, banyak digunakan jenis kontroler modern yang menggunakan mikrokontroler dengan logika tertentu, dengan *decoder* akan mengatur *switching* transistor sehingga terbentuk pola *switching* yang tepat pada tiap fase untuk mengelola akselerasi, kontrol kecepatan dan menyempurnakan efisiensi.

2.1.3 Prinsip Kerja Motor *Brushless* DC

Prinsip kerja mendasar dari motor BLDC adalah teori mengenai medan magnet, saat suatu kutub utara dengan medan magnet yang keluar akan saling tolak menolak dengan kutub yang sejenisnya begitupun sebaliknya akan saling tarik menarik jika magnetnya berlawan kutub. Dari prinsip sederhana diatas dapat diterapkan dalam penggunaan sistem kerja motor BLDC, yang memiliki medan magnet permanen pada *rotor* dan gaya elektromagnet (magnet yang ditimbulkan dari pemberian input arus listrik) pada bagian kumparan *stator*. Pada motor BLDC, fungsi pengaturan terhadap inputan arus yang harus diberikan ke kumparan stator untuk dapat menimbulkan medan elektromagnet yang tepat guna memutar rotor dilakukan oleh kontroler. Elemen inilah yang menjadi elemen utama yang membedakannya dengan motor DC konvensional, dan menggantikan kerja komutasi mekanisnya.

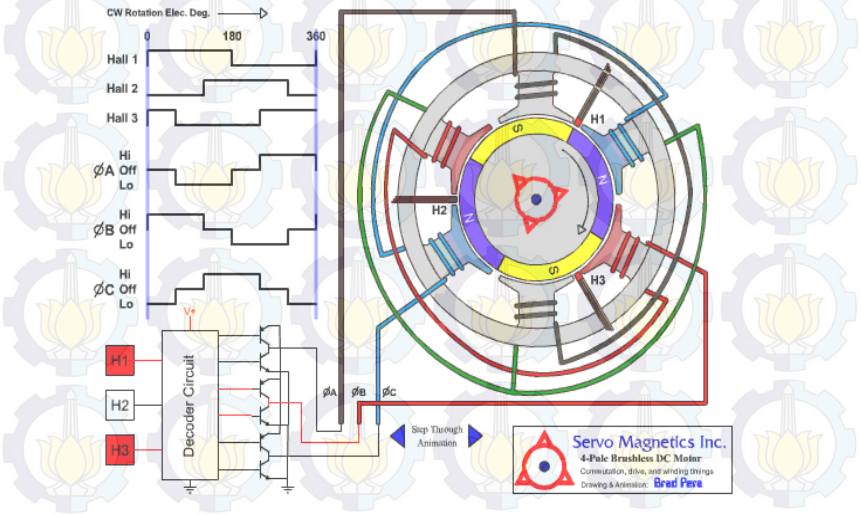
Gambar 2.5 memperlihatkan prinsip kerja motor BLDC 3 fasa dan 2 pasang kutub *rotor* secara umum. Terlihat pada gambar bahwa pendeteksian posisi *rotor* menggunakan 3 buah sensor *hall effect* H1, H2, dan H3 yang diletakkan pada ujung plat dan membentuk lingkaran dengan interval 120°. Pada gambar yang pertama, H2 mendeteksi fluks magnetik kutub utara dan *decoder* mengaktifkan B+ C-. Dalam kondisi ini, arus mengalir ke kumparan stator B dan membentuk kutub utara. Arus juga akan mengalir ke kumparan stator C dan membentuk kutub selatan. Akibatnya terjadi gaya tolak menolak antara *stator* dan *rotor* sehingga *rotor* akan berputar searah jarum jam. Begitu seterusnya kutub *rotor* akan berjalan sesuai dengan pensaklaran yang terus beganti. Dengan mengulang proses pensaklaran sesuai urutan seperti terlihat pada Gambar

2.5, maka *rotor* yang terdiri dari magnet permanen akan berputar secara terus menerus.

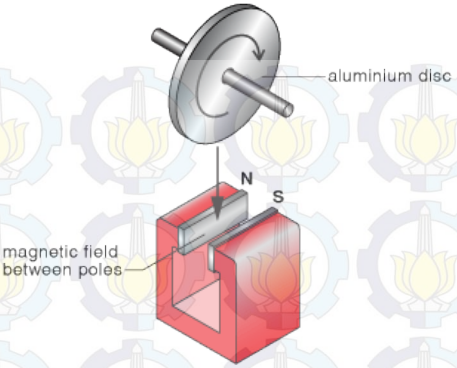
2.2 Rem Elektromagnetik

Sistem pengereman ini menggunakan gaya elektromagnetik yang timbul dari suatu magnet permanen tetap atau kumparan yang diberikan arus listrik untuk memperlambat suatu gerakan. Konstruksi dasarnya berupa suatu piringan logam non-ferromagnetik yang terpasang pada suatu poros yang berputar. Piringan logam tersebut diapit oleh kumparan yang dialiri arus listrik hingga menimbulkan medan magnet yang kutubnya saling berlawanan. Logam piringan tersebut akan memotong medan magnet yang ditimbulkan oleh kumparan tersebut sehingga menimbulkan *eddy current* atau arus *eddy*.

Arus *eddy* merupakan arus listrik yang timbul bilamana suatu piringan logam berada di sekitar medan magnet yang garis-garis gayanya sedang berubah-ubah. Arus *eddy* ini mempunyai medan magnet yang arahnya berlawanan dengan arah gerak piringan logam. Akibatnya laju piringan logam akan tertahan akibat dari adanya arus *eddy* ini. Gambar 2.6 akan memperlihatkan struktur dan konstruksi dari sebuah sistem rem elektromagnetik.



Gambar 2.5 Prinsip Kerja Motor BLDC. [6]



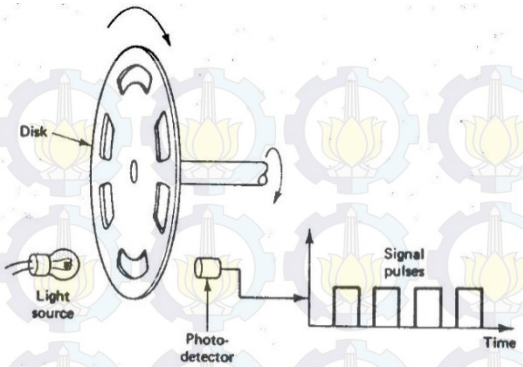
Gambar 2.6 Struktur dari Sebuah Rem Elektromagnetik. [7]

Rem elektromagnetik biasa diletakkan dekat dengan bagian yang bergerak. Rem ini bekerja pada kondisi yang dingin dan memenuhi persyaratan energi pengereman kecepatan tinggi karena tanpa adanya gesekan. Selain pada kendaraan listrik, aplikasi rem elektromagnetik juga dapat ditemukan pada sistem pengereman kereta api.

2.3 Sensor *Rotary Encoder*

Sensor *rotary encoder* adalah sebuah sensor elektromekanik yang dapat memonitor suatu gerakan atau keadaan posisi. Sensor ini umumnya menggunakan sensor optik untuk menghasilkan pulsa yang dapat diterjemahkan menjadi gerakan, posisi dan arah. Posisi sudut poros benda yang berputar dapat diolah menjadi informasi berupa kode digital oleh sensor *rotary encoder* untuk digunakan sebagai *feedback* dan diteruskan ke rangkaian kendali. [8]

Rotary encoder tersusun dari sebuah piringan tipis yang memiliki lubang pada sekeliling lingkaran. LED ditempatkan pada salah satu sisi piringan sehingga cahaya dapat menembus piringan melalui lubang-lubang tersebut. Di sisi yang lain, sebuah *phototransistor* ditempatkan persis bersebrangan dengan posisi LED. *phototransistor* ini akan menerima cahaya yang dipancarkan oleh LED tersebut dan akan diteruskan ke mikrokontroler sebagai pulsa. Untuk lebih jelasnya, konstruksi dan cara kerja sensor *rotary encoder* dapat dilihat pada Gambar 2.7.



Gambar 2.7 Konstruksi dan Prinsip Kerja Sensor *Rotary Encoder*. [9]

Piringan pada sensor *rotary encoder* dikopel dengan poros motor, sehingga apabila motor berputar, piringan tersebut akan ikut berputar pula. Apabila posisi piringan mengakibatkan cahaya dari LED dapat diterima oleh *phototransistor* melalui lubang-lubang yang ada, maka *phototransistor* akan mengalami saturasi dan menghasilkan suatu pulsa gelombang persegi. Banyaknya deretan pulsa yang dihasilkan *phototransistor* akan menentukan akurasi dari suatu *rotary encoder*. Akibatnya, semakin banyak lubang yang terdapat pada piringan *rotary encoder*, semakin baik pula akurasi dari *rotary encoder* tersebut.

2.4 Arduino Uno

Arduino Uno adalah *board* mikrokontroler berbasis Atmega328. Alat ini mempunyai 14 pin *input*/*output* digital dan 6 diantaranya dapat digunakan sebagai *output* PWM, 6 *input* analog, resonator keramik 19MHz, koneksi USB, soket listrik, ICSP *header*, dan tombol *reset* [1]. Dengan kabel USB alat ini mudah dihubungkan ke komputer dan dapat diaktifkan dengan baterai atau adaptor AC ke DC. Spesifikasi Arduino Uno adalah sebagai berikut :

- Mikrokontroler : Atmega328
- Arduino
- Tegangan operasi : 5 Volt
- Tegangan *input* : 7-12 Volt
- (direkomendasikan)
- Tegangan *input* : 6-20 Volt
- (batasan)

- Pin *input*/*output* digital : 14 (6 diantaranya *output* PWM)
- Pin *input* analog : 6
- Arus DC tiap pin I/O : 40mA
- Arus DC untuk pin 3,3 Volt : 50mA

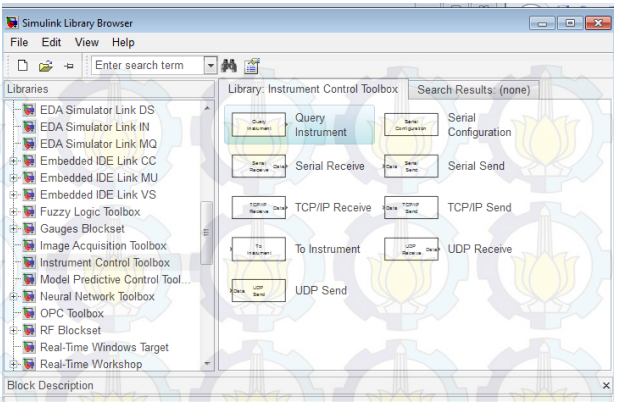
Pada Arduino Uno terdapat 6 *input* analog, dengan nama A0 hingga A5. Pin tersebut memiliki resolusi 10-bit (1024 nilai yang berbeda). *Input* analog ini berupa tegangan dari *ground* ke 5V, walaupun begitu terdapat kemungkinan untuk mengubah batas atas dan bawah dengan menggunakan pin AREF dan fungsi *analogReference()*.

2.5 MATLAB R2014a

MATLAB merupakan paket program dengan bahasa pemrograman yang tinggi untuk mengembangkan algoritma, visualisasi data, dan komputasi numerik. Program MATLAB ini dapat digunakan untuk menyelesaikan masalah komputasi dengan lebih cepat dibandingkan dengan bahasa pemrograman tradisional, seperti C, C++, dan Fortran. MATLAB digunakan untuk banyak aplikasi seperti *signal and image processing*, desain kontrol, pengujian dan pengukuran, permodelan, dan analisis. [1]

Simulink merupakan bagian dari MATLAB untuk memodelkan, mensimulasikan, dan menganalisa sistem dinamik. Simulink dapat membentuk model dari awal atau memodifikasi model yang sudah ada sesuai dengan apa yang diinginkan. Selain itu simulink juga mendukung sistem *linear* dan *non-linear*, permodelan waktu kontinyu atau diskrit, atau gabungan. Simulink ini dapat digunakan sebagai media untuk menyelesaikan masalah dalam industri nyata meliputi kedirgantaraan dan pertahanan, otomotif, komunikasi, elektronik dan pemrosesan sinyal,

Salah satu modul dalam simulink yang dapat digunakan untuk komunikasi perangkat keras adalah *Instrument Control Toolbox*, seperti terlihat pada Gambar 2.8. Modul ini merupakan kunpulan fungsi *m-file* yang dibangun pada lingkungan komputasi teknis MATLAB. *Toolbox* ini menyediakan kerangka kerja untuk komunikasi instrumen yang mendukung GPIB *interface*, standar VISA, TCP/IP, dan protokol UDP. *Toolbox* ini memperluas fitur dasar *serial port* yang ada dalam MATLAB. Selain itu *toolbox* ini berfugsi untuk komunikasi data antara *workspace* MATLAB dan peralatan lainnya. Data tersebut dapat berbentuk biner atau *text*.



Gambar 2.8 Dialog Tampilan *Instrument Control Toolbox.*

Komunikasi *serial* merupakan protokol dasar tingkat rendah untuk komunikasi antara dua peralatan atau lebih. Pada umumnya satu komputer dengan modem, printer, mikrokontroler, atau peralatan lainnya. *Serial port* mengirim dan menerima informasi *bytes* dengan hubungan seri. *bytes* tersebut dikirimkan menggunakan format biner atau karakter ASCII (*American Standard Code for Information Interchange*). Dalam komunikasi serial MATLAB, agar data ASCII dapat diproses *real time*, maka digunakan ASCII *encode* dan *decode* yang terdapat pada *xPC Target Library for RS232.* ASCII *encode* merupakan blok dalam simulink yang digunakan untuk mengubah data *bytes* menjadi karakter ASCII. Sedangkan ASCII *decode* merupakan blok simulink yang digunakan untuk mengubah karakter ASCII menjadi data *bytes* yang kemudian dapat dikonversi sesuai kebutuhan.

2.6 Identifikasi Sistem

Identifikasi sistem merupakan suatu langkah awal dalam menganalisa sistem dinamik. Menurunkan suatu fungsi alih yang baik dan sesuai merupakan salah satu bagian terpenting dalam proses menganalisa sebuah sistem secara keseluruhan. Fungsi alih yang baik dan sesuai cocok digunakan untuk analisa, prediksi, dan desain sistem, *regulator*, dan filter. Fungsi alih memiliki bentuk yang bermacam-macam. Salah satu bentuknya adalah *transfer function* yang cocok untuk permasalahan analisa transien dan sebuah sistem LTI (*Linear Time Invariant*) dan sistem dengan *Single-Input Single-Output* (SISO). Disisi lain, fungsi alih

dengan bentuk *state space* sangat cocok untuk menganalisa suatu sistem dengan *Multiple-Input Multiple-Output* (MIMO).

Fungsi alih dari sebuah sistem dapat diperoleh melalui dua cara yaitu permodelan fisik dan identifikasi sistem. Permodelan fisik merupakan pendekatan analitik berdasarkan hukum fisika seperti hukum newton dan hukum kesetimbangan. Permodelan ini menjelaskan dinamika dalam sistem. Yang kedua adalah identifikasi sistem. Hal ini dilakukan dengan pendekatan eksperimental. Model ini berdasarkan data dari eksperimen yang kemudian didapatkan nilai parameter sistem. Pada beberapa kasus yang sangat kompleks, sangat sulit untuk menentukan model berdasarkan pemahaman fisik. [10]

Identifikasi sistem pada suatu sistem dapat dilakukan dengan menggunakan metode identifikasi statis maupun dinamis. Identifikasi statis dilakukan untuk mendapatkan *gain* dan *time sampling* dari suatu sistem. Identifikasi statis pada suatu sistem dilakukan dengan memberikan suatu *input setpoint* yang bernilai konstan terhadap waktu. Sistem yang digunakan untuk identifikasi statis adalah sistem dengan *loop* terbuka tanpa kontroler. Sedangkan pada identifikasi dinamis, prosedur identifikasinya memiliki beberapa perbedaan mendasar. Diantaranya adalah perbedaan antara sinyal uji dan metode pemodelan yang digunakan. Jika identifikasi statis menggunakan *input setpoint* yang konstan terhadap waktu, maka identifikasi dinamis menggunakan sinyal acak atau *random* sebagai sinyal ujinya. Pada Tugas Akhir kali ini, digunakan metode identifikasi dinamis. Pemilihan metode ini disebabkan karena respon yang sangat cepat dari motor BLDC, sehingga sangat sulit untuk mengamati respon transien dari sistem jika menggunakan identifikasi statis.

2.6.1 Identifikasi Dinamis

Identifikasi dinamis dilakukan untuk mendapatkan pemodelan dari suatu sistem atau *plant*. Identifikasi dinamis memiliki beberapa kelebihan dibandingkan identifikasi statis. Pada identifikasi dinamis, input yang diberikan pada sistem memiliki frekuensi yang berubah-ubah, sehingga karakteristik dan sifat dari sistem dapat diteliti dengan lebih cermat. Input yang diberikan ini dinamakan sinyal *Pseudo-Random Binary Sequence*.

Setelah respon dari sistem diperoleh, langkah selanjutnya adalah melakukan pendekatan fungsi alih dari sistem berdasarkan respon yang didapat. Ada beberapa metode yang bisa dilakukan untuk mendapatkan fungsi alih dari sistem. Salah satunya adalah metode ARX atau *Auto-*

Regressive Exogeneous. Penjelasan selanjutnya akan diberikan pada paragraf dibawah ini.

2.6.1.1 Sinyal Pseudo-Random Binary Sequence

Terdapat beberapa perbedaan mendasar antara identifikasi statis dan dinamis. Diantaranya adalah perbedaan antara sinyal uji dan metode pemodelan yang digunakan. Jika identifikasi statis menggunakan input *setpoint* yang konstan terhadap waktu, maka identifikasi dinamis menggunakan sinyal acak atau *random* sebagai sinyal ujinya. Sinyal ini memiliki frekuensi yang berubah-ubah, sehingga memungkinkan karakteristik sistem dapat diketahui secara lebih teliti. Sinyal tersebut dinamakan sinyal *Pseudo-Random Binary Sequence* (PRBS). Sinyal PRBS mirip dengan bilangan acak secara nyata, tapi juga dapat disebut semu atau *pseudo* karena bersifat deterministik [10]. Sinyal PRBS dapat dihasilkan dari penggunaan *shift register*. Salah satu contoh sinyal uji PRBS dapat dilihat pada Gambar 2.9.

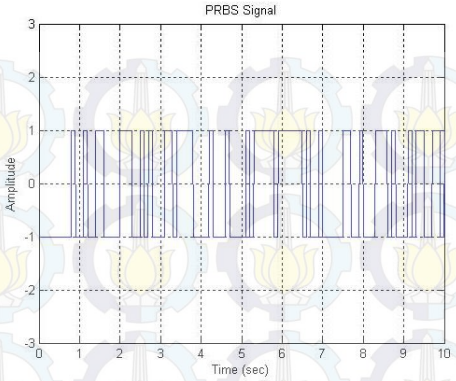
2.6.1.2 Auto-Regressive Exogeneous (ARX)

Langkah selanjutnya adalah menentukan pemodelan yang akan digunakan. Terdapat banyak macam pemodelan yang digunakan pada identifikasi dinamis, antara lain *Auto-Regressive* (AR), *Auto-Regressive Model with External Input* (ARX), *Auto-Regressive Moving Average* (ARMA), dan yang lainnya. Pada pembahasan Tugas Akhir kali ini, digunakan pendekatan ARX untuk mendapatkan fungsi alih yang diharapkan. ARX dipilih karena relatif sederhana dan cukup representatif [10]. Secara matematis, struktur pendekatan ARX dapat dituliskan sebagaimana Persamaan 2.1.

$$y(t)+a_1y(t-1)+...+a_{n_a}y(k-n_a)= \\ b_1u(t-n_k)+...+b_{n_b}u(t-n_b-n_k+1)+e(t) \tag{2.1}$$

dengan penjelasan variabel,

$y(t)$	:	*Output* pada waktu *t*
n_a	:	Banyaknya jumlah *pole*
n_b	:	Banyaknya jumlah *zero* ditambah satu
n_k	:	Nilai *dead time*. Jumlah masukan yang terjadi sebelum memberikan pengaruh pada keluaran.



Gambar 2.9 Sinyal Uji *Pseudo-Random Binary Sequence* (PRBS).

Secara ringkas, struktur pemodelan ARX dapat dituliskan sebagaimana Persamaan 2.2

$$A(q)y(t) = B(q)u(t-n_k) + e(t)$$
$$A(q) = 1 + a_1 q^{-1} + \ldots + a_{n_a} q^{-n_a} \quad (2.2)$$
$$B(q) = b_1 + b_2 q^{-1} + \ldots + a_{n_b} q^{n_b+1}$$

2.6.2 Validasi Model

Setelah fungsi alih didapat melalui proses identifikasi sistem, tahapan selanjutnya adalah tahapan validasi model. Validasi model diukur dengan cara mengukur nilai *error* setiap variabelnya. Tujuan dari validasi model adalah untuk mengukur apakah nilai pada pemodelan sudah mendekati dengan nilai sebenarnya dari sistem.

Dalam menggunakan proses validasi model, terdapat beberapa metode dan tolok ukur yang dapat digunakan. Diantaranya adalah metode *Root Mean Square Error* (RMSE). RMSE mengukur akurasi pada nilai deret waktu secara statistik seperti halnya regresi. RMSE dapat merepresentasikan ukuran dari *error* rata-rata karena RMSE membandingkan hasil data pengukuran dan data pemodelan pada skala yang sama antara kedua data tersebut. Formulasi perhitungan RMSE yang mempresentasikan nilai *error* dalam bentuk presentasi dapat dilihat pada Persamaan 2.3 dan Persamaan 2.4.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n} e_i^2}; \quad i = 1, 2, 3, ..., n \tag{2.3}$$

$$e_i = \frac{A_i - M_i}{A_i} \times 100\%; \quad i = 1, 2, 3, ..., n \tag{2.4}$$

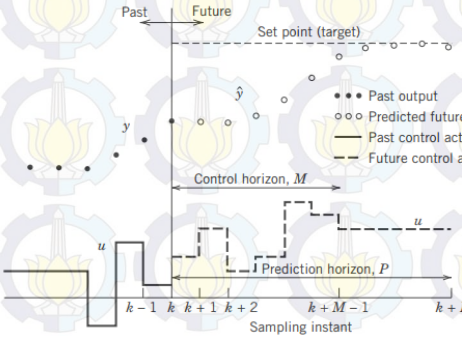
dengan penjelasan variabel,

N	:	Jumlah data
i	:	Urutan data
e_i	:	Nilai *error*
M_i	:	Nilai data hasil pemodelan
A_i	:	Nilai data hasil pengukuran

Semakin kecil nilai pengukuran RMSE suatu pemodelan, semakin baik pula model yang diberikan.

2.7 *Model Predictive Control* (MPC)

Tujuan utama dari sebuah *Model Predictive Control* (MPC) adalah untuk menghitung trayektori dari sinyal kontrol *u* (*manipulated variable*) yang akan datang untuk mengoptimalkan perilaku yang akan datang (*future behavior*) pada sinyal *output y* pada sebuah *plant* berdasarkan pada nilai pengukuran saat ini dan prediksi dari nilai *output* yang akan datang. Objektif dari kontroler MPC adalah untuk menentukan nilai sinyal kontrol (*sequence of control moves*) sehingga nilai *output* yang diprediksi akan mendekati nilai *setpoint* dengan optimal [11]. Konsep umum dari kontroler MPC dapat dilihat pada Gambar 2.10



Gambar 2.10 Konsep dari Kontroler *Model Predictive Control* [12]

Pada Gambar 2.10, dapat dilihat susunan dari nilai *output* saat ini (*actual output*) *y*, nilai *output* terprediksi (*predicted output*) $\hat{y}$, dan *manipulated input* atau sinyal kontrol *u*. Pada setiap waktu *sampling k*, kontroler MPC menghitung himpunan dari nilai *M* atau *control horizon* (selanjutnya disebut N_c) dari *input* $\{u(k+i-1), i=1,2,...,M\}$. Nilai *input* akan ditahan pada nilai konstan setelah *M* langkah didalam sinyal kontrol tersebut. Nilai *input* akan dihitung sedemikian sehingga nilai himpunan dari *P* keluaran atau *output* terprediksi $\{y(k+i), i=1,2,...,P\}$ akan mencapai nilai *setpoint* yang diinginkan. *P* merupakan nilai dari *prediction horizon* (selanjutnya disebut N_p) pada kontroler MPC. Perhitungan nilai kontrol pada kontroler MPC dihitung berdasarkan nilai optimal dari suatu fungsi objektif atau indeks performansi *J*. [12]

2.7.1 Model *State-Space* dengan *Embedded Integrator*

Pertama, kita mengasumsikan sebuah sistem *single-input single-output* yang dideskripsikan sebagai berikut:

$$\begin{aligned} x_m(k+1) &= A_m x_m(k) + B_m u(k) \\ y(k) &= C_m x_m(k) + \mathrm{D}_m u(k) \end{aligned} \tag{2.5}$$

Dimana *u* adalah variabel manipulasi atau variabel kontrol, x_m merupakan variabel *state* dan *y* adalah variabel *output*. Dikarenakan prinsip dari *receding horizon control*, dimana *state* saat ini dibutuhkan untuk menghitung prediksi dan kontrol, maka kita mengasumsikan bahwa *input u*(*k*) tidak dapat mempengaruhi *output y*(*k*) pada waktu yang sama. Oleh karena itu, nilai D_m dapat kita abaikan. Oleh karena itu, Persamaan 2.5 dapat ditulis kembali sebagai berikut:

$$\begin{aligned} x_m(k+1) &= A_m x_m(k) + B_m u(k) \\ y(k) &= C_m x_m(k) \end{aligned} \tag{2.6}$$

Untuk dapat menerapkan kontroler MPC pada suatu *plant*, maka kita perlu mengubah bentuk *state space* Persamaan 2.5 menjadi bentuk *augmented model*. Adapun bentuk *augmented model* dapat didefinisikan sebagai berikut:

$$\overbrace{\begin{bmatrix}\Delta x_m(k+1)\\ y(k+1)\end{bmatrix}}^{x(k+1)} = \overbrace{\begin{bmatrix}A_m & 0_m^T\\ C_m A_m & 1\end{bmatrix}}^{A}\overbrace{\begin{bmatrix}\Delta x_m(k)\\ y(k)\end{bmatrix}}^{x(k)} + \overbrace{\begin{bmatrix}B_m\\ C_m B_m\end{bmatrix}}^{B}\Delta u(k) \tag{2.7}$$

$$y(k) = \overbrace{\begin{bmatrix}0_m & 1\end{bmatrix}}^{C}\begin{bmatrix}\Delta x_m(k+1)\\ y(k+1)\end{bmatrix}$$

Adapun $o_m = \overbrace{\begin{bmatrix}0 & 0 & \cdots & 0\end{bmatrix}}^{n_1}$ dan matriks *A,B,C* biasa disebut *augmented model*. Model *state space* inilah yang akan digunakan selanjutnya untuk merancang sebuah kontroler MPC.

2.7.2 Perhitungan *Prediction Output* & *Future Control*

Setelah mendapatkan *augmented model*, langkah selanjutnya adalah menghitung nilai output terprediksi dan variabel kontrol yang akan datang. Variabel kontrol yang akan datang dapat ditulis sebagaimana Persamaan 2.8.

$$\Delta u(k_i),\ \Delta u(k_i+1),\ \ldots,\ \Delta u(k_i+N_c-1) \tag{2.8}$$

N_c merupakan nilai *control horizon*, yaitu jumlah langkah kontrol berkelanjutan yang diterapkan dan diprediksi oleh kontroler MPC dalam sebuah *sampling time*. Selain itu, variabel *output* terprediksi dapat diperkirakan dan diprediksi dalam jumlah sampel N_p, dimana N_p merupakan nilai *prediction horizon*. Adapun variabel *output* terprediksi dapat dituliskan dalam Persamaan 2.9.

$$x(k_i+1|k_i),\quad x(k_i+2|k_i),\ \ldots,\ x(k_i+N_p|k_i) \tag{2.9}$$

Sebagai catatan, nilai N_c harus lebih kecil atau sama dengan nilai N_p. Setelah itu, nilai *output* terprediksi dan variabel kontrol yang akan datang dapat dihitung dengan menggunakan Persamaan 2.10.

$$Y = Fx(k_i) + \Phi\Delta U \tag{2.10}$$

Matriks *F* dan *Φ* dapat diformulasikan sebagai berikut:

$$F = \begin{bmatrix} CA \\ CA^2 \\ CA^3 \\ \vdots \\ CA^{N_p} \end{bmatrix} ; \Phi = \begin{bmatrix} CB & 0 & \cdots & 0 \\ CAB & CB & \cdots & 0 \\ CA^2B & CAB & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ CA^{N_p-1}B & CA^{N_p-2}B & \cdots & CA^{N_p-N_c}B \end{bmatrix} \quad (2.11)$$

2.7.3 Indeks Performansi Kontroler MPC

Dalam sebuah kontroler MPC, diperlukan proses optimasi yang mempunyai objektif kontrol untuk meminimalkan *error* yang terbentuk dari selisih nilai referensi dengan nilai keluaran dari *plant*. Optimasi tersebut dilakukan dengan mendeskripsikan sebuah nilai dan parameter indeks performansi *J* yang merefleksikan objektif kontrol dari kontroler MPC. Indeks performansi tersebut dapat didefinisikan sebagai berikut:

$$J = (R_s - Y)^T (R_s - Y) + \Delta U^T \overline{R} \Delta U \quad (2.12)$$

$$R_s^T = \overbrace{\begin{bmatrix} 1 & 1 & \cdots & 1 \end{bmatrix}}^{N_p} r(k_i) \quad (2.13)$$

R_s merupakan vektor yang mempunyai informasi sinyal referensi atau *setpoint* yang didefinisikan seperti tertulis pada Persamaan 2.13. Persamaan $(R_s - Y)^T (R_s - Y)$ pada indeks performansi *J* di Persamaan 2.12 mempunyai tujuan untuk meminimalkan *error* yang terjadi antara nilai *predicted output* dengan *setpoint* yang diberikan pada sistem. Sedangkan persamaan $\Delta U^T \overline{R} \Delta U$ merefleksikan seberapa besar nilai ΔU yang akan dihasilkan ketika fungsi objektif indeks performansi *J* dibuat sekecil mungkin. Matriks $\overline{R}$ adalah matriks diagonal yang berbentuk $\overline{R} = r_w I_{N_c \times N_c} \ (r_w \geq 0)$ dan digunakan sebagai parameter *tuning* kontroler MPC. Variabel r_w merupakan *tuning parameter* untuk performa *closed-loop system* pada kontroler MPC. Pada kasus dimana nilai $r_w = 0$, indeks performansi *J* akan mempunyai objektif untuk meminimalkan nilai *error* $(R_s - Y)^T (R_s - Y)$ sekecil mungkin tanpa mempedulikan seberapa besar nilai ΔU yang akan dihasilkan oleh kontroler MPC. Untuk kasus dimana nilai r_w dibuat semakin besar, indeks performansi *J* pada Persamaan 2.12 akan diterjemahkan ke dalam situasi dimana kita akan meminimalkan

nilai *error* $(R_s - Y)^T (R_s - Y)$ secara hati-hati dengan mempertimbangkan seberapa besar nilai ΔU yang akan dihasilkan oleh kontroler MPC.

Untuk mendapatkan nilai kontrol optimal yang akan meminimalisasi indeks performansi *J*, kita dapat mengekspresikan indeks performansi *J* pada Persamaan 2.12 sebagai berikut:

$$\begin{aligned} J &= (R_s - Fx(k_i))^T (R_s - Fx(k_i)) - 2\Delta U^T \Phi^T \\ &\quad (R_s - Fx(k_i)) + \Delta U^T (\Phi^T \Phi + \bar{R}) \Delta U \end{aligned} \tag{2.14}$$

Setelah itu, Persamaan 2.14 diturunkan terhadap ΔU sebagai berikut:

$$\frac{\partial J}{\partial \Delta U} = -2\Phi^T (R_s - Fx(k_i)) + 2(\Phi^T \Phi + \bar{R}) \Delta U \tag{2.15}$$

Kondisi yang dibutuhkan untuk meminimalkan indeks performansi *J* dicari pada kondisi sebagai berikut:

$$\frac{\partial J}{\partial \Delta U} = 0 \tag{2.16}$$

Dari mensubtitusi Persamaan 2.15 dengan Persamaan 2.16, maka kita dapat menyimpulkan solusi optimal dari sinyal kontrol pada kontroler MPC sebagai Persamaan 2.17.

$$\Delta U = (\Phi^T \Phi - \bar{R})^{-1} (\Phi^T \bar{R}_s - \Phi^T Fx(k_i)) \tag{2.17}$$

Nilai $(\Phi^T \Phi - \bar{R})^{-1} \Phi^T R_s$ berhubungan dengan perubahan nilai *setpoint*, sedangkan nilai $-(\Phi^T \Phi - \bar{R})^{-1} \Phi^T F$ berhubungan dengan *state feedbeck control* pada kontroler MPC.

2.7.4 *Closed-loop Control System*

Meskipun nilai optimal parameter pada vektor ΔU mengandung sinyal kontrol $\Delta u(k_i)$, $\Delta u(k_i + 1)$, …, $\Delta u(k_i + N_c - 1)$, kita hanya dapat mengimplementasikan sampel pertama dari urutan atau *sequence* tersebut, contoh $\Delta u(k_i)$, dan mengabaikan nilai atau urutan selanjutnya. Prinsip ini disebut dengan *Receding Horizon Control* (RHC). Ketika periode sampling selanjutnya datang, nilai pengukuran yang paling baru diambil dari *state* vektor $x(k_i+1)$ untuk penghitungan sinyal kontrol yang

baru. Prosedur ini terus berlanjut pada kondisi *real time* untuk memenuhi prinsip RHC. Oleh karena itu, sinyal kontrol yang didapatkan dari Persamaan 2.17 dapat ditulis ulang sebagaimana Persamaan 2.18.

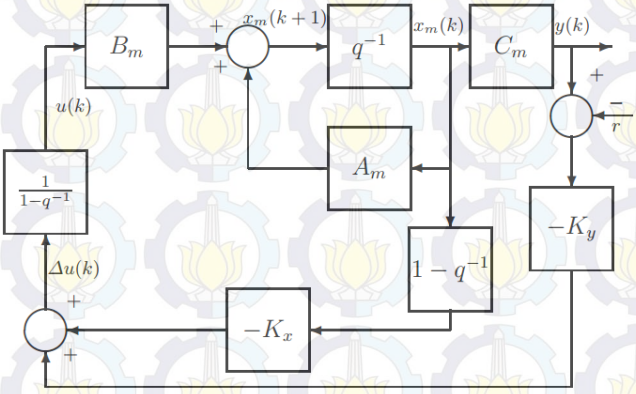
$$\begin{aligned}\Delta u(k_i) &= \overbrace{\begin{bmatrix}1 & 0 & \dots & 0\end{bmatrix}}^{N_c}(\Phi^T\Phi - \overline{R})^{-1}(\Phi^T\overline{R}_s - \Phi^T F x(k_i)) \\ &= K_y r(k_i) - K_{MPC}x(k_i)\end{aligned} \tag{2.18}$$

Nilai K_{MPC} merupakan baris pertama dari $(\Phi^T\Phi + \overline{R})^{-1}(\Phi^T F)$. Sedangkan nilai K_y pada Persamaan 2.18 merupakan baris pertama dari $(\Phi^T\Phi + \overline{R})^{-1}(\Phi^T\overline{R}_s)$. Persamaan 2.18 merupakan bentuk standar dari *linear time-invariant state feedback control*. Nilai *gain* atau penguatan pada *state feedback control* tersebut adalah K_{MPC}.

Untuk mencari persamaan sistem *closed-loop* pada kontroler MPC, digunakan *augmented model* seperti terlihat pada Persamaan 2.19.

$$x(k+1) = Ax(k) + B\Delta u(k) \tag{2.19}$$

Sistem *closed-loop* seperti pada Gambar 2.11 dapat dicari dengan mensubtitusi Persamaan 2.18 kedalam Persamaan 2.19 dan mengganti indeks k_i ke dalam k sebagaimana Persamaan 2.20.



Gambar 2.11 Diagram Blok Sistem Kontrol *Loop* Tertutup *Model Predictive Control* untuk Waktu Diskrit.

$$\begin{aligned} x(k+1) &= Ax(k) - BK_{MPC}x(k) + BK_y r(k) \\ &= (A - BK_{MPC})x(k) + BK_y r(k) \end{aligned} \tag{2.20}$$

Nilai *eigenvalues* pada sistem *closed-loop* diatas dapat ditentukan dengan mengamati persamaan karakteristik *closed-loop* sebagai berikut:

$$\det\left[\lambda I - (A - BK_{MPC})\right] = 0 \tag{2.21}$$

Dikarenakan struktur unik dari matriks *A* dan *C*, kolom terakhir dari matriks *F* identik dengan matriks $\overline{R}_s$, yaitu $\begin{bmatrix}1 & 1 & \dots & 1\end{bmatrix}^T$. Oleh karena itu, *gain* K_y identik dengan elemen terakhir pada *gain* K_{MPC}. Perlu dicatat bahwa nilai vektor *state variable* $x(k_i) = \begin{bmatrix}\Delta x_m(k)^T & y(k)\end{bmatrix}^T$ dan dengan definisi K_y, kita dapat mendeklarasikan bahwa *gain* $K_{MPC} = \begin{bmatrix}K_x & K_y\end{bmatrix}$, dimana K_x mendefinsikan hubungan *feedback gain vector* dengan $x_m(k)$. Sedangkan K_y mendefinisikan hubungan antara *feedback gain* dengan $y(k)$. Oleh karena itu, blok diagram sistem *closed-loop* pada kontroler MPC dapat dilihat seperti pada Gambar 2.1. Notasi q^{-1} merupakan notasi *backward shift operator* dan $\frac{1}{1-q^{-1}}$ menotasikan *discrete-time integrator*.

2.7.5 *State Estimation*

Pada perancangan kontroler MPC, kita mengasumsikan bahwa nilai *state* $x(k_i)$ selalu tersedia setiap waktu k_i dan mengasumsikan bahwa semua variabel *state* dapat terukur. Akan tetapi, pada kondisi yang sebenarnya, tidak semua variabel *state* dapat diukur, beberapa diantaranya mustahil untuk diukur. Oleh karena itu, diperlukan sebuah pendekatan untuk mengatasi masalah ini. Salah satu diantaranya adalah dengan mengestimasi variabel $x(k)$ dari pengukuran suatu proses. Sistem yang digunakan untuk mengestimasi variabel yang tidak diketahui ini dinamakan *observer*. Struktur diagram blok untuk kontroler MPC dengan menggunakan *observer* dapat dilihat pada Gambar 2.12.

Pada dasarnya, nilai *state* variabel $x(k_i)$ dapat diestimasi nilainya menggunakan sebuah *observer* dalam bentuk:

$$\hat{x}(k_i+1) = \overbrace{A\hat{x}(k_i) + B\Delta u(k_i)}^{model} + \overbrace{K_{ob}(y(k_i) - C\hat{x}(k_i))}^{correction\ term} \tag{2.22}$$

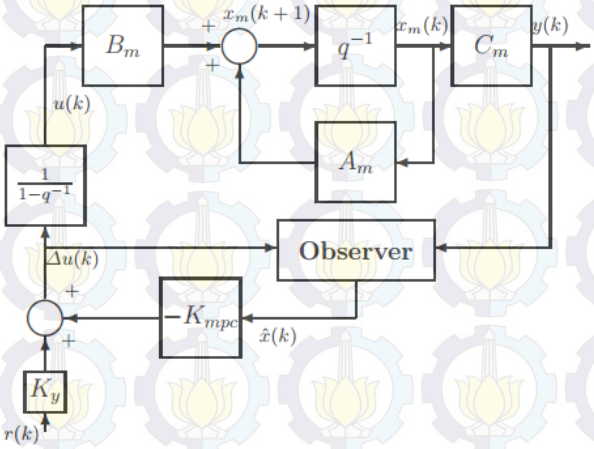
Gain K_{ob} merupakan nilai dari *observer gain matrix*. Pada Persamaan 2.22, terlihat bahwa bagian pertama merupakan *original model* dari sistem, sedangkan bagian kedua merupakan *correction term* yang berdasarkan pada *error* antara nilai *output* yang diukur dengan nilai *output* terprediksi yang menggunakan nilai estimasi $\hat{x}_m(k)$.

Perlu diingat bahwa dalam implementasi kontroler MPC menggunakan *observer*, harus digunakan matriks (*A*,*B*,*C*) dalam bentuk *augmented model*. Dengan mengubah nilai $\hat{x}(k_i)$ menggantikan $x(k_i)$, indeks performansi dapat kita rubah sebagaimana Persamaan 2.23

$$\begin{aligned} J &= (R_s - F\hat{x}(k_i))^T(\bar{R}_s r(k_i) - F\hat{x}(k_i)) + 2\Delta U^T \Phi^T \\ &\quad (R_s - F\hat{x}(k_i)) + \Delta U^T(\Phi^T\Phi + \bar{R})\Delta U \end{aligned} \tag{2.23}$$

Solusi kontrol optimal ΔU dapat diambil menggunakan Persamaan 2.24 sebagai berikut:

$$\Delta U = (\Phi^T\Phi - \bar{R})^{-1}(\Phi^T\bar{R}_s - \Phi^T F x(k_i)) \tag{2.24}$$



Gambar 2.12 Diagram Blok Sistem Kontrol *Loop* Tertutup *Model Predictive Control* untuk Waktu Diskrit dengan menggunakan *Observer*.

Dengan mengaplikasikan prinsip *Receding Horizon Control* (RHC), dapat kita ambil solusi optimal $\Delta u(k_i)$ pada setiap waktu k_i sebagaimana Persamaan 2.25 berikut.

$$\Delta u(k_i) = K_y r(k_i) - K_{MPC}\, x(k_i) \qquad (2.25)$$

Persamaan 2.25 merupakan hukum standar *state feedback control* dengan nilai estimasi $x(k_i)$.

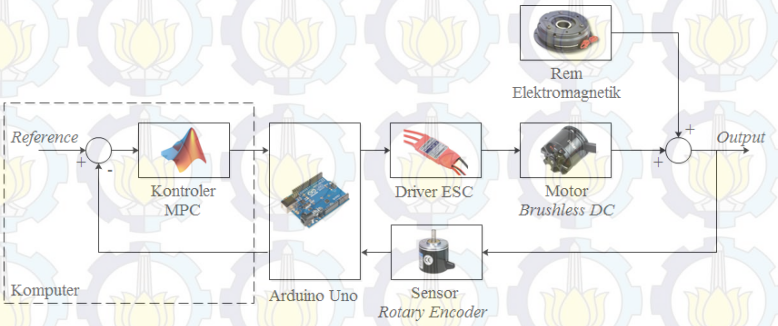
BAB 3
PERANCANGAN SISTEM

Bab 3 pada laporan Tugas Akhir ini akan menjelaskan mengenai perancangan sistem dari pengaturan kecepatan motor BLDC. Selanjutnya, Bab 3 akan menjelaskan mengenai identifikasi dan pemodelan dari sistem. Terakhir, pada bab ini juga akan dijelaskan mengenai perancangan kontroler *Model Predictive Control* pada sistem.

3.1 Gambaran Umum Sistem

Pada pelaksanaan Tugas Akhir kali ini, akan digunakan sebuah *plant* atau sistem yang terdiri dari komponen utama berupa motor *Brushless DC* (BLDC). Pengerjaan *plant* atau sistem ini dikerjakan oleh 5 orang yang kami sebut dengan BLDC Team. *Plant* atau sistem ini kami beri nama BLDC-V1. Selain itu, digunakan rem elektromagnetik untuk memberikan efek pembebanan pada motor BLDC. Pembebanan ini digunakan untuk mensimulasikan kondisi jalan sebenarnya yang akan dilalui oleh kendaraan listrik. Gambar 3.1 menunjukkan blok diagram dari perancangan sistem yang digunakan pada Tugas Akhir kali ini.

Selain perangkat keras berupa motor BLDC dan rem elektromagnetik, ditambahkan pula beberapa komponen pendukung, seperti *driver* untuk motor BLDC dan rem elektromagnetik. Untuk komponen pengukuran kecepatan, digunakan sensor *rotary encoder* yang digunakan sebagai umpan balik yang akan diteruskan ke dalam komputer.



Gambar 3.1 Blok Diagram Sistem Pengaturan Kecepatan Motor *Brushless DC* (BLDC).

Dalam sistem ini, motor BLDC merupakan komponen yang dikontrol untuk mencapai *output* yang diinginkan. Demi tujuan tersebut, digunakan kontroler berbasis *Model Predictive Control* (MPC) guna mencapai performansi yang diinginkan. Kontroler ini akan mengeluarkan sinyal kontrol yang kemudian dikirimkan ke *driver* untuk menggerakkan motor BLDC. Dengan begitu, *output* yang dihasilkan diharapkan bisa sesuai dengan referensi yang diinginkan.

3.2 Perancangan Perangkat Keras

Pada tahap perancangan keras, terdapat 2 jenis perancangan yang dilakukan, yaitu perancangan mekanik dan elektronik. Perancangan mekanik merupakan perancangan komponen utama pada *plant*, yaitu berupa motor BLDC dan rem elektromagnetik. Sedangkan perancangan elektronik merupakan perancangan untuk kontroler, *driver* dan rangkaian sensor yang akan digunakan pada *plant* ini. Kontroler akan diprogram didalam PC dan sebagai perantara komputer dengan *plant*, digunakan mikrokontroler Arduino yang juga berfungsi sebagai perangkat akuisisi data.

Arduino akan menerima data dari sensor dan mengirimkannya kepada komputer. Selain itu, digunakan digunakan pula rangkaian *driver* untuk menggerakan motor BLDC dan memberi *input* arus pada rem elektromagnetik. Terdapat pula rangkaian sensor yang berfungsi mengukur *input* arus yang masuk ke dalam rem elektromagnetik dan sensor kecepatan motor BLDC berupa sensor *rotary encoder*.

3.2.1 Perancangan Mekanik

Pada *plant* sistem pengaturan kecepatan motor BLDC, terdapat beberapa komponen utama yang digunakan, antara lain motor BLDC sebagai tenaga penggerak dan objek yang dikontrol. Selain itu, terdapat rem elektromagnetik yang digunakan untuk memberikan efek pembebanan pada motor BLDC. Rem elektromagnetik diberikan *input* arus DC yang berbentuk PWM (*Pulse Width Modulation*). Untuk lebih jelasnya, penjelasan mengenai konstruksi motor BLDC dan rem elektromagnetik dapat dilihat pada subbab dibawah ini.

3.2.1.1 Motor Brushless DC

Motor BLDC yang digunakan merupakan motor BLDC yang digunakan pada pesawat *aeromodelling*. Motor BLDC jenis ini merupakan miniatur dari motor BLDC penggerak kendaraan listrik

dikarenakan konstruksinya yang lebih kecil. Motor yang digunakan merupakan berasal dari produsen RCTimer dengan nomor seri HP2212-1000KV. Bentuk fisik motor BLDC yang digunakan pada *plant* ini dapat dilihat pada Gambar 3.2. Sedangkan spesifikasi motor BLDC dapat dilihat pada Tabel 3.1.

3.2.1.2 Electronic Speed Control (ESC)

Electronic Speed Control atau yang biasa disingkat ESC merupakan *driver* atau *inverter* yang mengubah arus DC dalam bentuk PWM menjadi tegangan AC yang dapat digunakan untuk menggerakkan motor BLDC. ESC biasa digunakan pada *quadcopter* untuk menggerakkan motor BLDC.

Tabel 3.1 Spesifikasi Motor BLDC RCTimer HP2212-1000KV. [13]

Parameter	Nilai	
Kekuatan Magnet	40 µH	
Berat Motor	59,06 gram	
KV	1002,7 RPM/Volt	
Kecepatan Motor	Tanpa Beban	11.130 RPM
	Beban Maksimal	6.840 RPM
Input Arus	Tanpa Beban	0,7 Ampere
	Beban Maksimal	13,9 Ampere
Power Motor	Beban Minimal	44,4 Watt
	Beban Maksimal	154,29 Watt
Input Baterai	Lithium Polimer 2S-4S	

Gambar 3.2 Motor *Brushless* DC Tipe *Outrunner* dengan Tipe RCTimer HP2212-1000KV. [13]

Input yang diberikan sebagai masukan ke dalam ESC ini sebesar 20 ms atau 50 Hz. Untuk kecepatan minimal, nilai sinyal PWM yang masuk ke dalam ESC bernilai 1 ms. Sedangkan untuk kecepatan maskimal, dibutuhkan sinyal PWM sebesar 2 ms untuk kecepatan hingga 13.000 RPM. Sinyal PWM ini akan diolah oleh *timer* dan *mikrokontroler* yang ada pada ESC untuk mengaktifkan rangkaian *power transistor* yang terdiri dari beberapa MOSFET. Rangkaian *power transistor* inilah yang memberikan *input* arus ke dalam motor BLDC berupa sinyal sinusoidal 3 fasa dengan perbedaan sudut sebesar 120°.

Pada Tugas Akhir kali ini, digunakan ESC Turnigy PLUSH-25A. ESC ini sanggup memberi arus pada motor BLDC hingga 25 Ampere. Untuk lebih jelasnya, spesifikasi dan bentuk dari ESC Turnigy PLUSH-25A dapat dilihat pada Tabel 3.2 dan Gambar 3.3 berikut.

Gambar 3.3 *Electronic Speed Control* Turnigy PLUSH-25A. [14]

Tabel 3.2 Spesifikasi ESC Turnigy PLUSH-25A. [14]

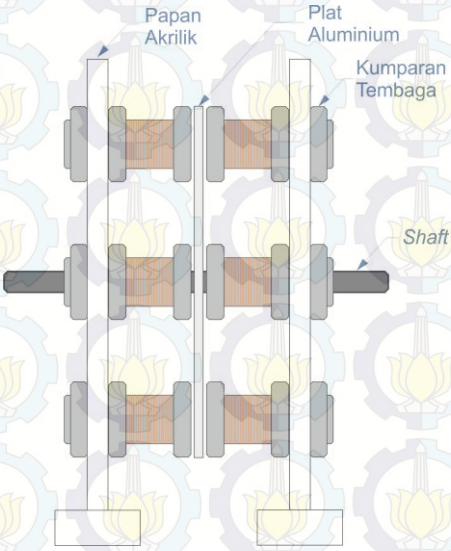
Parameter	Nilai	
Model	Turnigy PLUSH-25A	
Continous Current	25 Ampere	
Burst Current (>10s)	35 Ampere	
Kecepatan Motor	2 *pole*	210.000 RPM
	6 *pole*	70.000 RPM
	12 *pole*	35.000 RPM
Input Baterai	Lithium Polimer 2S-4S	
Dimensi	22 gram	
Berat	45 mm x 24 mm x 11 mm	

3.2.1.3 Rem Elektromagnetik

Alat ini digunakan untuk memberikan efek pembebanan pada motor BLDC. Bentuk perancangan rem elektromagnetik yang digunakan pada *plant* ini dapat dilihat pada Gambar 3.4. Medan elektromagnetik dari rem ini dihasilkan oleh beberapa kumparan yang dihubungkan secara seri dan diberikan masukan arus DC. Arus DC yang masuk ke dalam rem elektromagnetik berbentuk PWM yang *duty cycle*-nya diatur oleh *driver* dan Arduino. Adapun sumber tegangannya didapat dari jala jala PLN yang disearahkan melalui rangkaian *rectifier*.

Rem elektromagnetik ini disusun dari 8 kumparan yang disusun secara seri. 8 kumparan ini dipisahkan menjadi 2 bagian dan diantara celah tersebut dipasang piringan aluminium. Piringan aluminium tersebut lalu dipasang ke dalam sebuah *shaft* yang selanjutnya dikopel dengan motor BLDC. Perbandingan *gear* yang digunakan untuk mengkopel motor BLDC dengan shaft tersebut adalah 1:4.

Konstruksi rem sendiri dari 4 kumparan pada tiap sisinya. Tegangan input yang diberikan pada tiap sisi sebesar 36 Volt. Jumlah lilitan tiap kumparan dihitung berdasarkan rumus berikut ini:



Gambar 3.4 Perancangan Konstruksi Fisik Rem Elektromagnetik.

$$N = \frac{44}{d} \times V = \frac{44}{1,4} \times 36 = 1131 \quad (3.1)$$

N : Jumlah lilitan tiap sisi
d : Diameter kumparan (cm)
V : Tegangan input (Volt)

Jumlah tersebut masih harus dibagi 4 dikarenakan tiap sisi rem yang mempunyai 4 kumparan. Jadi, tiap kumparan memiliki 283 lilitan.

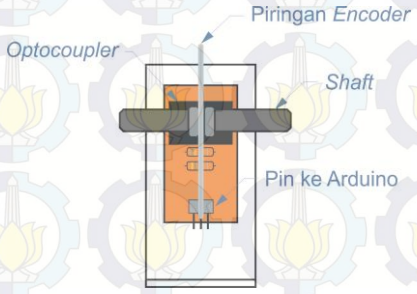
3.2.2 Perancangan Elektronik

Perancangan elektronik ini meliputi desain *layout* rangkaian PCB serta pengkabelan. Rangkaian elektronik pada *plant* ini meliputi rangkaian *driver* rem elektromagnetik, rangkaian penyearah gelombang AC atau *rectifier* serta rangkaian sensor kecepatan *rotary encoder* dan sensor arus. Selain itu, digambarkan pula rancangan pengkabelan pada mikrokontroler Arduino.

3.2.2.1 Rangkaian Sensor Rotary Encoder

Rotary encoder merupakan salah satu jenis sensor yang biasa digunakan untuk mengukur kecepatan. Sensor ini dipasang pada *shaft* seperti yang terlihat seperti Gambar 3.5. Sensor terdiri dari piringan besi yang mempunyai lubang dengan jumlah dan sudut yang tertentu.

Sensor *rotary encoder* terdiri dari rangkaian *optocoupler* yang tersusun atas *Light Emitting Diode* atau (LED) dan *receiver phototransistor* seperti terlihat pada Gambar 3.6. Apabila *optocoupler* tidak terhalang apapun, LED akan memancarkan cahaya dan akan diterima oleh *phototransistor* sehingga menghasilkan pulsa.

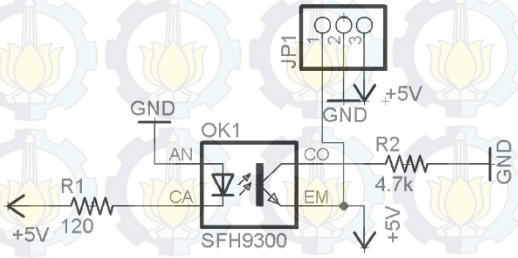


Gambar 3.5 Sensor *Rotary Encoder* yang Terpasang pada *Shaft*.

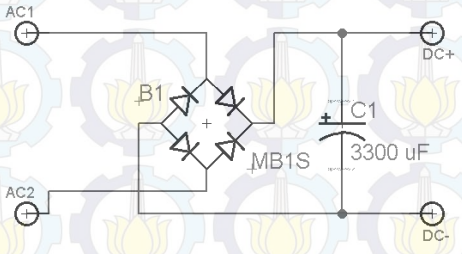
Tetapi, ketika *optocoupler* terhalang, cahaya dari LED tidak bisa diterima oleh *phototransistor* sehingga tidak menghasilkan pulsa apapun. Untuk lebih jelasnya, rangkaian penyusun sensor *rotary encoder* ini dapat dilihat pada Gambar 3.6

3.2.2.2 Rangkaian Penyearah Gelombang AC

Untuk menjalankan komponen rem elektromagnetik, dibutuhan suatu tegangan DC yang dialirkan pada kumparan sehingga menghasilkan medan elektromagnetik. Oleh karena itu, diperlukan suatu rangkaian yang mengubah tegangan AC yang berasal dari jala-jala listrik PLN dan mengubahnya menjadi tegangan DC. Rangkaian tersebut dinamakan rangkaian penyearah gelombang AC atau yang biasa dikenal dengan nama rangkaian *rectifier*. Bentuk skematik dari rangkaian *rectifier* dapat dilihat pada Gambar 3.7. Rangkaian *rectifier* tediri dari 4 buah *dioda* (*diode bridge*) yang disusun seperti pada Gambar 3.7.



Gambar 3.6 Skema Rangkaian *Rotary Encoder* yang Tersusun dari *Optocoupler* dan Resistor.

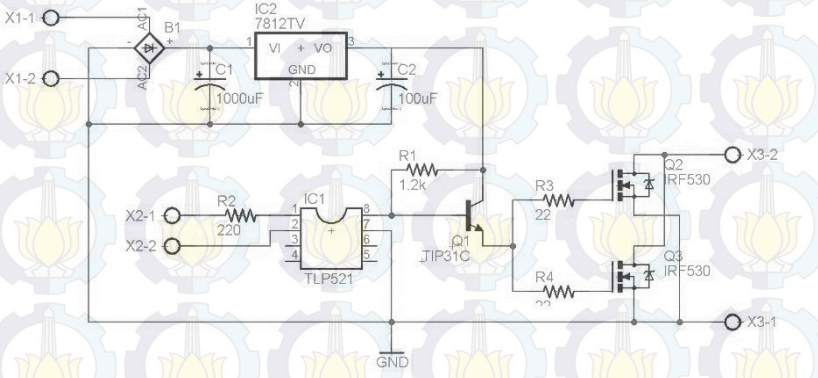


Gambar 3.7 Rangkaian Penyearah Gelombang AC yang Terdiri dari *Diode Bridge* dan Kapasitor.

Sifat dari dioda yang unik, yaitu *short circuit* ketika *forward bias* dan *open circuit* ketika *reverse bias* dimanfaatkan untuk mengubah tegangan AC menjadi tegangan DC. Setelah keluar dari *diode bridge* tegangan DC tidak dapat langsung dialirkan ke dalam kumparan. Ini dikarenakan tegangan DC tersebut masih memiliki tegangan riak atau *ripple* yang dapat merusak komponen. Untuk meminimalisir riak tersebut, digunakanlah sebuah kapasitor yang berfungsi sebagai filter dan menghilangkan tegangan *ripple* tersebut.

3.2.2.3 Rangkaian Driver Rem Elektromagnetik

Rem elektromagnetik sangat berperan penting dalam kesulurahan sistem ini. Rem digunakan untuk memberikan pembebanan pada motor BLDC. Untuk mengatur nilai pembebanan yang akan diberikan kepada motor BLDC, dibutuhkan suatu rangkaian yang digunakan untuk mengatur besaran arus DC yang masuk ke dalam kumparan. Oleh karena itu, dibuatlah suatu *driver* yang digunakan untuk mengatur arus yang masuk ke dalam kumparan. Arus ini berupa arus DC yang telah disearahkan oleh rangkaian penyearah. Pengaturan arus yang masuk ke dalam kumparan tersebut dilakukan menggunakan teknik PWM atau *Pulse Width Modulation* yang diatur menggunakan mikrokontroler Arduino. Rangkaian skematik *driver* rem elektromagnetik dapat dilihat pada Gambar 3.8. Sumber tegangan pada rangkaian driver ini berasal dari tegangan AC.

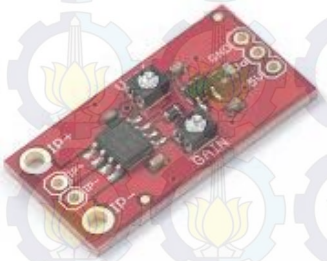


Gambar 3.8 Skema Rangkaian *Driver* Rem Elektromagnetik.

Pada prinsipnya, *driver* rem ini mempunyai kemiripan dengan *driver* yang biasa digunakan untuk menjalankan motor DC. *Driver* ini berkerja dengan diberikan tegangan 12V untuk mengaktifkan komponen komponen seperti regul*a*tor (IC7812), *transistor* (TIP31C), dan *optocoupler* (TLP521). Cara kerjanya, ketika mikrokontroler memberikan tegangan dengan *duty cycle low stage*, maka terdapat beda potensial antara mikrokontroler dengan *regulator* sehingga *optocoupler* akan aktif. Saat itu, *transistor* tidak akan aktif karena tegangan dialirkan oleh *transistor* yang ada pada *optocoupler*. Saat mikrokontoler memberikan tegangan dengan *duty cycle high stage*, tidak ada beda potensial antara *regulator* dan mikrokontroler sehingga *optocoupler* tidak aktif. Karena itu tegangan 12 V akan mengalir melalui basis dari *transistor* dan mengaktifkan *transistor* tersebut. Tegangan dari kolektor mengalir ke emitor dan akan mengaktifkan basis dari dua MOSFET (IRF530) yang dipasang paralel. Basis pada kedua MOSFET aktif, maka tegangan supply akan dapat mengalir ke kedua MOSFET tersebut dan mengaktifkan rem elektromagnetik.

3.2.2.4 Rangkaian Sensor Arus

Sensor arus digunakan untuk mengukur besaran nilai arus yang dialirkan ke dalam rem elektromagnetik. Besaran arus ini sangat penting untuk diketahui sebab akan digunakan sebagai parameter dan nilai pembebanan yang akan diberikan kepada motor. Sensor arus yang digunakan berjenis sensor ACS712 buatan produsen SparkFun. Sensor ini dapat mengukur arus dengan batas maksimal mencapai 5 Ampere. Tabel 3.3 akan menjelaskan dengan sekilas mengenai spesifikasi dari sensor arus ini. Gambar dari sensor arus yang digunakan pada Tugas Akhir ini dapat dilihat pada Gambar 3.9.



Gambar 3.9 Sensor Arus SparkFun ACS712 5A Breakout

Tabel 3.3 Spesifikasi Sensor Arus SparkFun ACS712 5A. [15]

Parameter	Nilai
Model	SparkFun ACS712 5A Breakout
Arus Maksimal	5 Ampere
Bandwith	80 kHz
Sensitivitas *Output*	66-185 mV/A
Tegangan Operasi	5 Volt
Tegangan *Output*	2,5-5 Volt

3.2.2.5 Mikrokontroler Arduino

Pada *plant* kali ini, digunakan mikrokontroler Arduino sebagai alat akuisisi data yang menjembatani antara *software* dan *aktuator* beserta sensor. Arduino digunakan karena pemrogramannya yang terbilang sederhana dan cukup fleksibel. Adapun jenis Arduino yang digunakan pada pelaksanaan Tugas Akhir kali ini adalah jenis Arduino Uno R3 seperti terlihat pada Gambar 3.10

Arduino Uno memiliki 14 pin *input/output* yang mana 6 pin dapat digunakan sebagai *output* PWM, 6 analog *input*, *crystal* osilator 16 MHz, koneksi USB, *jack power*, kepala ICSP, dan tombol *reset*. Pengiriman data dari sistem minimum Arduino ke PC/laptop dilakukan melalui koneksi USB. Bahasa yang digunakan oleh perangkat Arduino ini yaitu menggunakan bahasa C+ untuk pemrogramannya. *Software* Arduino ini dilengkapi dengan kumpulan *library* yang cukup lengkap, sehingga dapat membantu pengguna dalam penggunaannya. Sistem minimum Arduino Uno R3 secara lengkap dapat dilihat pada Gambar 3.10.

Gambar 3.10 Mikrokontroler Arduino Uno R3. [16]

3.3 Perancangan Perangkat Lunak

Perangkat lunak diperlukan dalam perancangan sistem sebagai *interface* antara *plant* dan komputer. Perangkat lunak yang digunakan dalam pengerjaan Tugas Akhir ini antara lain adalah penggunaan perangkat lunak Arduino dan MATLAB. Arduino digunakan sebagai alat akuisisi data, sedangkan MATLAB digunakan untuk membaca nilai dari sensor untuk keperluan identifikasi sistem dan mengirimkan sinyal kontrol pada ESC. Selain itu, MATLAB digunakan pula sebagai *software* untuk desain, simulasi dan implementasi kontroler.

3.3.1 *Software* Arduino

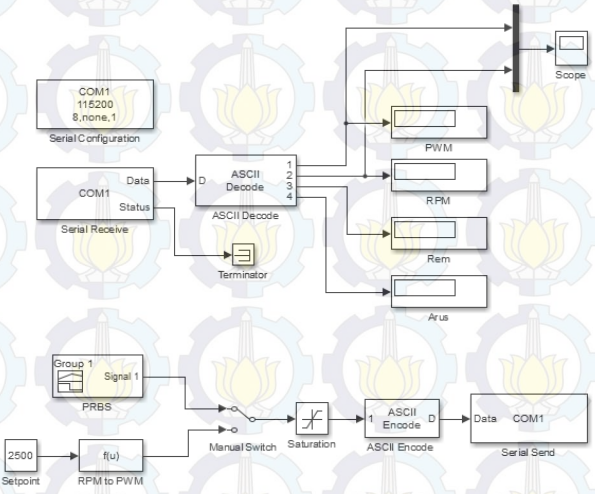
Seperti telah yang dijelaskan sebelumnya, mikrokontroler Arduino digunakan sebagai alat akuisisi data dari berbagai macam komponen pada *plant*. *Software* yang sering digunakan untuk meng-*compile* bahasa pemrograman pada Arduino bernama Arduino IDE. Program akuisisi data yang dikehendaki ditulis pada Arduino IDE di komputer, lalu di-*compile* dan di-*upload* menuju mikrokontroler via serial USB. Dalam Tugas Akhir kali ini, mikrokontroler Arduino secara garis besar digunakan untuk menerima data kecepatan dari sensor *rotary encoder* melalui pin 7, mengendalikan besaran PWM pada *driver* rem elektromagnetik melalui pin A0, menerima data dari sensor arus pada pin A2 dan mengirimkan *throttle* atau sinyal kontrol berupa PWM pada ESC pada pin 9. Tampilan program Arduino IDE dapat dilihat pada Gambar 3.11.

3.3.2 Software MATLAB

Selain Arduino IDE, *software* lain yang digunakan pada Tugas Akhir kali ini adalah *software* MATLAB. Versi MATLAB yang digunakan pada pelaksanaan Tugas Akhir kali ini mempunyai versi 2015a. *Software* MATLAB merupakan *software* yang sangat vital dan umum digunakan untuk desain kontroler beserta implementasinya. Pada *software* MATLAB, terdapat sub-program lainnya yang bernama Simulink. *Software* Simulink digunakan sebagai *Human Machine Interface* pada proses pengiriman dan penerimaan data melalui serial USB dari mikrokontroler Arduino seperti terlihat pada Gambar 3.12.

Selain itu, pada *software* Simulink juga terdapat blok-blok yang dapat digunakan untuk mendesain dan mengimplementasikan kontroler. Disisi lain, Simulink juga digunakan untuk proses identifikasi sistem *open loop* maupun *closed loop* menggunakan blok *Instrument Control Toolbox* melalui blok *Serial Send* dan *Serial Receive*.

Gambar 3.11 Tampilan Arduino IDE



Gambar 3.12 Blok Identifikasi Sistem *Open Loop* pada Simulink.

3.4 Identifikasi dan Pemodelan Sistem

Identifikasi merupakan sebuah proses yang harus dilewati guna mendapatkan parameter dan bentuk matematika dari sebuah sistem. Pada proses ini, identifikasi sistem dilakukan dengan memberikan beberapa nilai pembebanan yang diberikan pada motor BLDC. Setelah didapatkan respon dalam beberapa parameter pembebanan, respon tersebut dapat dicari fungsi alihnya melalui identifikasi dinamis.

3.4.1 Metode Pembebanan *Plant*

Pada Tugas Akhir kali ini, motor BLDC diberikan beban berupa rem elektromagnetik. Beban ini menggambarkan beban awal pada kendaraan elektrik yang ditenagai oleh motor BLDC. Pembebanan dilakukan dengan 3 nilai yang berbeda, yaitu beban minimal, nominal dan maksimal. Potensiometer digunakan untuk mengatur pemberian *input* PWM ke *driver* rem elektromagnetik melalui pin A0. Skala yang digunakan mulai dari nol untuk menon-aktifkan rem hingga skala 1000 untuk pemberian *input* maksimal pada rem. Nilai tegangan yang masuk ke dalam rem elektromagnetik dapat dilihat pada Tabel 3.4.

3.4.2 Metode Identifikasi dan Pemodelan

Identifikasi pada motor BLDC dilakukan pada sistem *open loop* menggunakan identifikasi dinamis. Identifikasi dinamis dilakukan dengan memberikan sinyal acak atau *random*. Sinyal ini biasa disebut dengan sinyal PRBS (*Pseudo-Random Binary Sequence*). Adapun input yang diberikan berupa nilai pulsa PWM yang dimasukkan ke dalam ESC. Pengambilan data dilakukan dengan membaca nilai sensor *rotary encoder* yang telah dihubungkan ke dalam Arduino. Mikrokontroler Arduino lalu dihubungkan ke dalam komputer melalui komunikasi *serial* untuk dibaca nilai *input* PWM dan *output* kecepatannya.

Setelah respon berhasil didapatkan, respon tersebut lalu dimodelan dengan pendekatan ARX pada orde 1 sampai dengan 3 menggunakan bantuan *System Identification Toolbox* pada *software* MATLAB. Lalu, model yang digunakan adalah model dengan nilai *error* yang paling kecil.

Tabel 3.4 Metode Pembebanan pada Sistem.

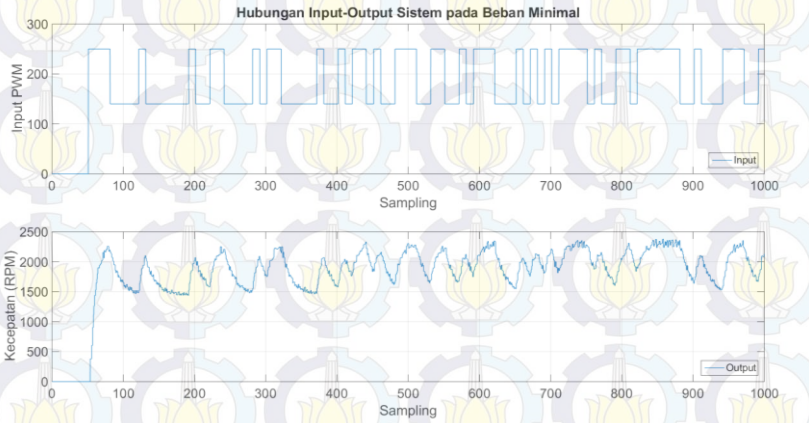
Metode Pembebanan	*Input* PWM	Nilai
Minimal	400	4,4 Volt
Nominal	600	7,89 Volt
Maksimal	800	12,56 Volt

3.4.3 Pemodelan Motor *Brushless* DC

Pemodelan motor BLDC didapatkan dari identifikasi dinamis melalui sistem *open loop*. Dalam hal ini dilakukan 3 jenis pemodelan, yaitu pemodelan beban minimal, nominal dan maksimal. Data yang telah didapatkan kemudian dianalisis menggunakan bantuan *System Identification Toolbox* pada *software* MATLAB untuk mendapatkan fungsi alih sistem melalui pendekatan ARX. Untuk memperjelas, Gambar 3.13 menyajikan perbandingan data mengenai hubungan *input-output* sistem identifikasi dinamis.

3.4.3.1 Beban Minimal

Pada beban minimal, motor BLDC diberikan pembebanan rem elektromagnetik pada tegangan 4,4 Volt. Setelah mengetahui respon dari sistem, barulah dilakukan pemodelan pada sistem melalui pendekatan ARX. Pemodelan sistem dilakukan hingga orde 3. Identifikasi dinamis untuk pemodelan ARX pada orde 1 sampai orde 3 pada beban minimaldapat diamti pada Tabel 3.5 berikut ini.

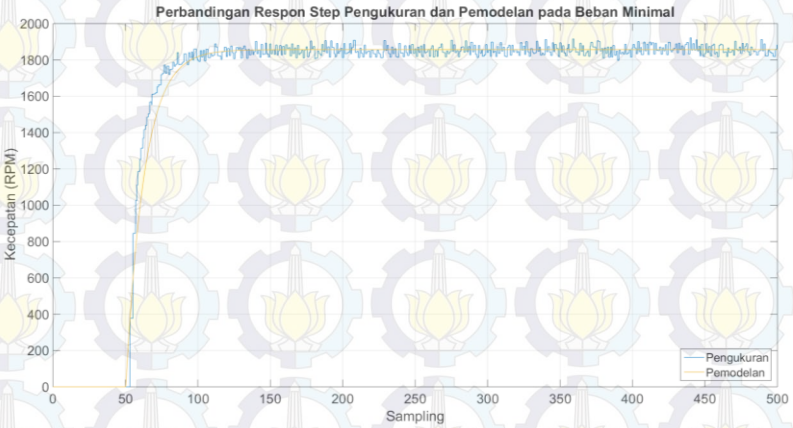


Gambar 3.13 Hubungan *Input-Output* Sistem pada Identifikasi Dinamis.

Gambar 3.14 menunjukkan perbandingan antara hasil respon pengukuran dengan respon pemodelan saat sistem diberikan sinyal masukan berupa sinyal *unit step*.

Tabel 3.5 Identifikasi Dinamis pada Beban Minimal.

Orde	Fungsi alih	RMSE (%)
1	$\frac{0,007196}{z-0,9992}$	11,7
2	$\frac{0,002513z+0,004715}{z^2-0,9957z-0,003521}$	6,98
3	$\frac{0,002525z^2-6,218\times10^{-15}z+0.004736}{z^3-0,9957z^2-1,846\times10^{-15}z-0.003537}$	7,38



Gambar 3.14 Perbandingan Respon Hasil Pengukuran dan Pemodelan pada Beban Minimal

3.4.3.2 Beban Nominal

Pada beban nominal, motor BLDC diberikan beban berupa rem elektromagnetik pada tegangan 7,77 Volt. Setelah mengetahui respon dari sistem, barulah dilakukan pemodelan pada sistem melalui pendekatan ARX. Pemodelan sistem dilakukan hingga orde 3. Identifikasi dinamis untuk pemodelan ARX pada orde 1 sampai orde 3 pada beban nominal dapat diamti pada Tabel 3.6 berikut ini. Identifikasi dinamis untuk pemodelan ARX pada orde 1 sampai orde 3 pada beban nominal dapat diamati pada Tabel 3.6 berikut ini.

Tabel 3.6 Identifikasi Dinamis pada Beban Nominal.

Orde	Fungsi alih	RMSE (%)
1	$\frac{0,007487}{z-0,9991}$	9,59
2	$\frac{0,003239z+0,004284}{z^2-0,9956z-0,003566}$	5,68
3	$\frac{0,003254z^2+9,887\times10^{-17}z+0.004303}{z^3-0,9955z^2-7,971\times10^{-15}z-0.003583}$	7,96

3.4.3.3 Beban Maksimal

Pada pembebanan terakhir, yaitu pada kondisi beban maksimal, motor akan diberikan beban dari rem elektromagnetik yang diberi input tegangan sebesar 12,56 Volt. Setelah mengetahui respon dari sistem, barulah dilakukan pemodelan pada sistem melalui pendekatan ARX. Pemodelan sistem dilakukan hingga orde 3. Identifikasi dinamis untuk pemodelan ARX pada orde 1 sampai orde 3 pada beban maksimal dapat diamati pada Tabel 3.7 berikut ini.

Tabel 3.7 Identifikasi Dinamis pada Beban Maksimal.

Orde	Fungsi alih	RMSE (%)
1	$\frac{0,009326}{z-0,9988}$	5,89
2	$\frac{0,002823z+0,006564}{z^2-0,9936z-0,005205}$	5,43
3	$\frac{0,002843z^2-2,475\times10^{-15}z+0.006606}{z^3-0,9935z^2-6,35\times10^{-15}z-0.005239}$	8,19

3.4.4 Pengujian dan Validasi

Setelah dilakukan pemodelan menggunakan pendekatan ARX, langkah selanjutnya adalah memvalidasi fungsi alih tersebut. Semakin baik suatu pemodelan, maka respon yang dihasilkan akan semakin menyerupai respon aslinya. Keakuratan suatu pemodelan dapat dilihat dari nilai *error*-nya, dalam hal ini dikalkulasi menggunakan metode *Root Mean Square Error* (RMSE). Semakin kecil nilai RMSE, semakin baik pula pemodelan yang dihasilkan.

Tabel 3.8 Pemodelan Motor BLDC pada Berbagai Macam Kondisi Pembebanan

Pembebanan	Fungsi alih	RMSE (%)
Minimal	$\frac{0,002513z + 0,004715}{z^2 - 0,9957z - 0,003521}$	6,98
Nominal	$\frac{0,003239z + 0,004284}{z^2 - 0,9956z - 0,003566}$	5,68
Maksimal	$\frac{0,002823z + 0,006564}{z^2 - 0,9936z - 0,005205}$	5,43

Berdasarkan hasil pemodelan diatas, kita dapat menyimpulkan bahwa pemodelan ARX dengan orde 2 mempunyai nilai *error* yang paling rendah untuk semua kondisi pembebanan. Oleh karena itu, pemlihan fungsi alih pada semua kondisi pembebanan yang akan digunakan pada simulasi dan implementasi dapat dilihat pada Tabel 3.8.

3.5 Perancangan Kontroler *Model Predictive Control*

Setelah fungsi alih dari sistem telah didapatkan, langkah selanjutnya adalah merancang kontroler untuk masing-masing pembebanan. Kontroler digunakan untuk mengembalikan respon ke nilai *setpoint* yang diinginkan sekalipun motor BLDC diberi beban. Pada perancangan kontroler ini, akan digunakan fungsi alih sistem pada pembebanan maksimal dikarenakan fungsi alih ini memiliki nilai RMSE paling kecil. Tahapan desain kontroler ini meliputi perancangan fungsi alih pada model *state-space*, desain *augmented model*, perancangan *state estimator* pada sistem menggunakan *observer* dan penentuan parameter kontroler *Model Predictive Control* (MPC).

3.5.1 Perancangan Model *State Space*

Kontroler MPC merupakan kontroler berbasis model. Artinya, diperlukan sebuah pemodelan fungsi alih yang baik agar kontroler yang telah didesain dapat bekerja secara optimal. Fungsi alih yang digunakan merupakan fungsi alih motor BLDC pada pembebanan maksimal. Fungsi alih ini tidak bisa langsung digunakan untuk merancang kontroler MPC, melainkan harus diubah terlebih dahulu ke dalam bentuk *state-space*.

Pada pembebanan maksimal, representasi fungsi alih yang bernilai RMSE paling kecil adalah sebagai berikut:

$$\frac{Y(z)}{U(z)} = \frac{b_0 z^2 + b_1 z + b_2}{z^2 + a_1 z + a_2} = \frac{0,002823z + 0,006564}{z^2 - 0,9936z - 0,005205} \quad (3.2)$$

Berdasarkan fungsi alih pada Persamaan 3.2, kita dapat menyimpulkan variabel-variabel sebagai berikut:

$a_1 = -0,9936$
$a_2 = -0,005205$
$b_0 = 0$
$b_1 = 0,002823$
$b_2 = 0,006564$

Bentuk *state-space* yang digunakan pada Tugas Akhir ini adalah bentuk *controllable canonical form* [17]. Bentuk *state-space* untuk orde 2 tersebut mempunyai struktur sebagai berikut:

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -a_2 & -a_1 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(k)$$
$$y(k) = \begin{bmatrix} b_2 - a_2 b_0 & b_1 - a_1 b_0 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + b_0 u(k) \quad (3.3)$$

Berdasarkan bentuk diatas, maka model *state-space* dari Persamaan 3.1 dapat kita simpulkan sebagai berikut:

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0,005205 & 0,9936 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(k)$$
$$y(k) = \begin{bmatrix} 0,006564 & 0,002823 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix}$$

3.5.2 Desain *Augmented Model*

Selanjutnya adalah mengubah bentuk *state-space* dari sistem ke dalam bentuk *augmented model* dari model diatas sebagai berikut:

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \end{bmatrix} = \overbrace{\begin{bmatrix} 0 & 1 \\ 0,005205 & 0,9936 \end{bmatrix}}^{A_m} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + \overbrace{\begin{bmatrix} 0 \\ 1 \end{bmatrix}}^{B_m} u(k)$$

$$y(k) = \overbrace{\begin{bmatrix} 0,006564 & 0,002823 \end{bmatrix}}^{C_m} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix}$$

Berdasarkan Persamaan 2.7, kita dapat mengubah bentuk *state-space* diatas menjadi bentuk *augmented model.*

$$\overbrace{\begin{bmatrix} \Delta x_m(k+1) \\ y(k+1) \end{bmatrix}}^{x(k+1)} = \overbrace{\begin{bmatrix} A_m & 0_m^T \\ C_m A_m & 1 \end{bmatrix}}^{A} \overbrace{\begin{bmatrix} \Delta x_m(k) \\ y(k) \end{bmatrix}}^{x(k)} + \overbrace{\begin{bmatrix} B_m \\ C_m B_m \end{bmatrix}}^{B} \Delta u(k)$$

$$y(k) = \overbrace{\begin{bmatrix} 0_m & 1 \end{bmatrix}}^{C} \begin{bmatrix} \Delta x_m(k+1) \\ y(k+1) \end{bmatrix}$$

$$\overbrace{\begin{bmatrix} \Delta x_m(k+1) \\ y(k+1) \end{bmatrix}}^{x(k+1)} = \overbrace{\begin{bmatrix} 0 & 1 & 0 \\ 0,005205 & 0,9936 & 0 \\ 0 & 0,0094 & 1 \end{bmatrix}}^{A} \overbrace{\begin{bmatrix} \Delta x_m(k) \\ y(k) \end{bmatrix}}^{x(k)} + \overbrace{\begin{bmatrix} 0 \\ 1 \\ 0,002823 \end{bmatrix}}^{B} \Delta u(k)$$

$$y(k) = \overbrace{\begin{bmatrix} 0 & 0 & 1 \end{bmatrix}}^{C} \begin{bmatrix} \Delta x_m(k+1) \\ y(k+1) \end{bmatrix}$$

Matriks *A,B,C* yang merupakan bentuk *augmented model* akan digunakan selanjutnya dalam merancang sebuah kontroler MPC.

3.5.3 Penentuan Parameter dan *Gain* Kontroler MPC

Langkah selanjutnya dalam perancangan kontroler MPC adalah menentukan parameter dari kontroler MPC. Parameter yang dimaksud adalah *prediction horizon* (N_p), *control horizon* (N_c) dan *tuning parameter* pada indeks performansi (r_w). Pada perancangan kali ini, kita akan menggunakan parameter *prediction horizon* sebesar 5 langkah, *control horizon* senilai 2 langkah dan *tuning* parameter indeks performansi sebesar 1. Berdasarkan Persamaan 2.10 dan 2.11, nilai *output* terprediksi dan variabel kontrol yang akan datang dapat dihitung dengan menggunakan persamaan berikut:

$$Y = Fx(k_i) + \Phi\Delta U$$

Matriks F dan Φ dapat diformulasikan sebagai berikut:

$$F = \begin{bmatrix} CA \\ CA^2 \\ CA^3 \\ \vdots \\ CA^{N_p} \end{bmatrix} ; \Phi = \begin{bmatrix} CB & 0 & \cdots & 0 \\ CAB & CB & \cdots & 0 \\ CA^2B & CAB & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ CA^{N_p-1}B & CA^{N_p-2}B & \cdots & CA^{N_p-N_c}B \end{bmatrix}$$

Setelah nilai parameter kontroler ditentukan, maka kita dapat menentukan nilai Matriks F dan Φ sebagai berikut:

$$F = \begin{bmatrix} CA \\ CA^2 \\ CA^3 \\ CA^4 \\ CA^5 \end{bmatrix} ; \Phi = \begin{bmatrix} CB & 0 \\ CAB & CB \\ CA^2B & CA^1B \\ CA^3B & CA^2B \\ CA^4B & CA^3B \end{bmatrix} \quad (3.4)$$

Langkah selanjutnya adalah memasukkan nilai matriks *augmented model* ke dalam Persamaan 3.3. Hasil yang didapatkan dapat dilihat pada persamaan dibawah ini.

$$F = \begin{bmatrix} 0 & 0,0094 & 1 \\ 0,0001 & 0,0187 & 1 \\ 0,0001 & 0,0280 & 1 \\ 0,0002 & 0,0373 & 1 \\ 0,0002 & 0,0466 & 1 \end{bmatrix}; \Phi = \begin{bmatrix} 0,0028 & 0 \\ 0,0122 & 0,0028 \\ 0,0215 & 0,0122 \\ 0,0308 & 0,0215 \\ 0,0401 & 0,0308 \end{bmatrix}$$

Setelah mendapatkan matriks F dan Φ dapat, parameter selanjutnya yang akan dicari adalah *gain* dari kontroler MPC. *Gain* tersebut adalah K_{MPC} dan K_y. Untuk mencari nilai K_{MPC}, terlebih dahulu kita harus mencari nilai dari matriks $(\Phi^T\Phi + \bar{R})^{-1}(\Phi^T F)$.

$$Y = (\Phi^T\Phi + \bar{R})^{-1}(\Phi^T F) \quad (3.5)$$

$$Y = \begin{bmatrix} 0 & 0,0039 & 0,107 \\ 0 & 0,0026 & 0,0607 \end{bmatrix}$$

Nilai *gain* K_{MPC} merupakan baris pertama dari matriks *Y*. Oleh karena itu, nilai *gain* K_{MPC} dapat kita simpulkan sebagai matriks berikut ini:

$$K_{MPC} = \begin{bmatrix} 0 & 0,0039 & 0,107 \end{bmatrix}$$

Setelah *gain* K_{MPC} ditemukan, langkah selanjutnya adalah mencari *gain* K_y. Penguatan atau *gain* ini dapat ditemukan dari nilai matriks $(\Phi^T \Phi + \overline{R})^{-1}(\Phi^T \overline{R_s})$.

$$Z = (\Phi^T \Phi + \overline{R})^{-1}(\Phi^T \overline{R_s}) \qquad (3.6)$$

$$Z = \begin{bmatrix} 0,107 \\ 0,607 \end{bmatrix}$$

Gain K_y merupakan baris pertama dari matriks *Z*. Berdasarkan matriks diatas, dapat kita ambil bahwa nilai K_y mempunyai nilai:

$$K_y = 0,107$$

3.5.4 Desain *State Estimation* Menggunakan *Observer*

Penggunaan *observer* pada sistem ini merupakan suatu langkah yang krusial. Ini dikarenakan adanya *state* yang tidak terukur pada sistem. Untuk mengatasinya, diperlukan sebuah *state estimation* berupa *observer* untuk mengestimasi nilai dari *state* yang tidak terukur ini.

Sebelum merancang sebuah *observer*, harus kita perhatian dahulu apakah sistem termasuk sistem *observability*. Artinya, semua *state* pada waktu $x(t_o)$ dapat ditentukan nilainya dari suatu *output* pada interval waktu tertentu [18]. Untuk menguji apakah suatu sistem *observable* atau tidak, sistem perlu diuji menggunakan matriks pada Persamaan 3.7 berikut.

$$rank\begin{bmatrix} C^T & A^T C^T & (A^T)^2 C^T \end{bmatrix} = n \qquad (3.7)$$

Variabel *n* merupakan jumlah *state* yang ada pada sistem. Pada Persamaan 3.5, matriks sistem yang digunakan adalah matriks yang

berdasarkan pada *augmented model*. Berdasarkan perhitungan pada MATLAB, nilai *rank* matriks diatas bernilai 3, sama dengan jumlah *state* pada *augmented model*. Dapat disimpulkan bahwa sistem *observable* dan dapat ditentukan setiap nilai *state*-nya pada waktu $x(t_o)$.

Setelah menentukan bahwa sistem *observable*, langkah selanjutnya adalah mendesain *observer*. Perlu diketahui, bahwa pada perancangan *observer*, matriks yang digunakan adalah matriks yang berasal dari *augmented model*. Jenis *observer* yang digunakan adalah *full state order observer*. Oleh karena itu, perlu ditemukan nilai K_e yang merupakan *gain observer*. *Gain observer* ini dapat ditemukan melalui beberapa metode, salah satunya metode *Ackermann* seperti yang digunakan pada Tugas Akhir ini. Pencarian nilai *gain observer* K_e menggunakan metode *Ackermann* dapat dilihat pada Persamaan 3.8 dan Persamaan 3.9.

$$K_e = \phi(A)\begin{bmatrix} C \\ CA \\ CA^2 \end{bmatrix}^{-1}\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{3.8}$$

$$\phi(A) = (s-\mu_1)(s-\mu_2)(s-\mu_3) \tag{3.9}$$

Variabel μ merupakan nilai *eigenvalue* yang diinginkan pada sistem. Dengan metode *trial-and-error*, *pole* yang diinginkan berada pada -0,001; -0,12 dan -0,03. Sehingga dengan bantuan software MATLAB, kita dapat menentukan nilai *gain observer* sebesar:

$$K_e = \begin{bmatrix} 121,9463 \\ 122,1254 \\ 2,1446 \end{bmatrix}$$

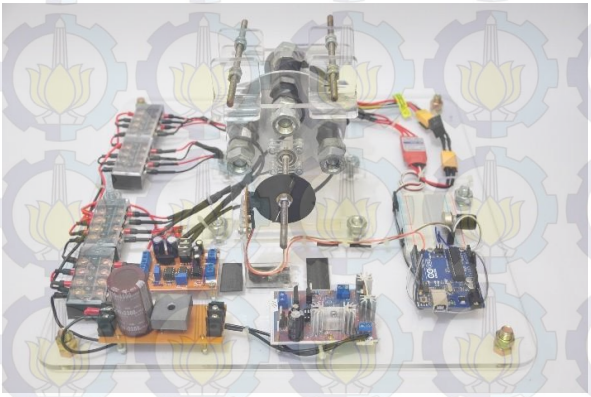
BAB 4
PENGUJIAN DAN ANALISA

Bab 4 menjelaskan tentang pengujian sistem berupa simulasi menggunakan software MATLAB dengan menggunakan pemodelan yang sudah didapatkan. Setelah mengetahui hasil simulasi, kontroler MPC akan diimplementasikan ke dalam sistem dan hasilnya akan dianalisa dengan menggunakan parameter karakteristik respon.

4.1 Gambaran Umum Pengujian Sistem

Pada pengerjaan Tugas Akhir ini, akan dilakukan beberapa pengujian pada sistem dengan beberapa kondisi pembebanan. Perancangan sistem seperti telah dijelaskan pada Bab 3 dapat direalisasikan menjadi sebuah sistem *plant* pengaturan kecepatan motor BLDC seperti terlihat pada Gambar 4.1. Sebelumnya, akan terlebih dahulu dilakukan pengujian terhadap beberapa komponen sistem. Setelah itu, akan dilakukan pengujian *open loop* pada motor BLDC dengan rentang kecepatan 1500-2500 RPM.

Tahapan selanjutnya adalah melakukan simulasi dari kontroler MPC. Simulasi akan dilakukan pada *software* MATLAB dengan bantuan *software* Simulink. Fokus utama dalam simulasi sistem ini adalah mengaplikasikan kontroler MPC pada sistem dengan objektif kontrol berupa *tracking* dengan nilai *setpoint* yang berubah-ubah.



Gambar 4.1 Realisasi *Plant* atau Sistem Pengaturan Kecepatan Motor *Brushless DC.*

Langkah terakhir dalam pengujian sistem adalah implementasi kontroler MPC pada sistem sesungguhnya. Akan dilakukan beberapa analisa dengan membandingkan respon yang didapat antara simulasi dan implementasi. Selain itu, akan dilakukan juga analisa pengaruh antara sebelum dan sesudah pemberian kontroler MPC pada sistem.

4.2 Pengujian Perangkat Keras

Pengujian perangkat keras pada pelaksanan Tugas Akhir kali ini bertujuan untuk mengetahui apakah perangkat keras yang berjalan pada sistem dapat bekerja secara semestinya. Pengujian perangkat keras tersebut meliputi pengujian *driver* motor BLDC berupa *Electronic Speed Control* (ESC).

4.2.1 Pengujian *Electronic Speed Control* Motor BLDC

Driver motor BLDC atau yang biasa disebut *Electronic Speed Control* (ESC) merupakan salah satu perangkat keras yang mempuyai kinerja paling krusial dalam menggerakkan motor BLDC. Oleh karena itu, penting untuk mengetahui dan memastikan ESC memiliki performa yang baik untuk menjalakan motor BLDC dengan baik.

Seperti yang telah dijelakan pada Bab 3, *Input* yang diberikan sebagai masukan ke dalam ESC ini mempunyai frekuensi sebesar 50 Hz atau 2 ms (*milisecond*). Untuk kecepatan minimal, nilai sinyal PWM yang masuk ke dalam ESC bernilai 1 ms. Sedangkan untuk kecepatan maskimal, dibutuhkan sinyal PWM sebesar 2 ms. Rentang nilai 1-2 ms tersebut di *map* atau diberi rentang baru bernilai 0-1000. Ini bertujuan untuk memudahkan pemberian *input* nilai atau masukan sinyal kontrol pada saat simulasi maupun implementasi. Nilai tegangan dari *input* PWM ini juga dihitung untuk mengetahui karakteristik dari motor BLDC. Hubungan *input* PWM, tegangan *output* driver dalam satuan milivolt dan kecepatan motor BLDC dapat dilihat pada Tabel 4.1

Berdasarkan Tabel 4.1 dapat kita dapat melihat bahwa hubungan tegangan *output* ESC linear dengan *input* PWM yang diberikan. Oleh karena itu, dapat disimpulkan bahwa *driver* ESC dapat digunakan untuk menggerakkan motor BLDC.

Tabel 4.1 Hubungan Nilai *Input* PWM, Tegangan *Output* ESC dan Kecepatan Motor BLDC.

Input PWM	Tegangan (mV)	Kecepatan Motor BLDC (RPM)
50	277,6	340
100	286,2	1099
150	294,2	1722
200	302,5	2196
250	310,8	2503
300	319,2	2751
350	327,3	2905
400	335,7	3043
450	343,9	3115
500	352,2	3199
550	360,2	3236
600	368,6	3300
650	376,7	3319
700	385,2	3363
750	393,3	3370
800	402,1	3409
850	410,2	3411
900	418,5	3471
950	426,8	3512
1000	435,4	3529

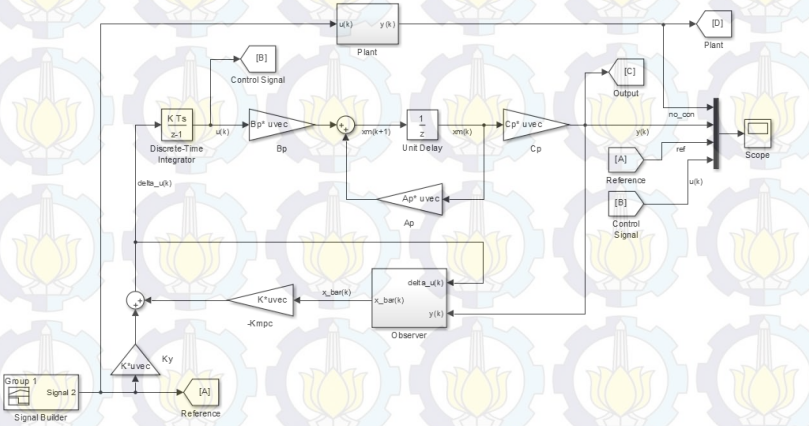
4.3 Simulasi Sistem

Simulasi merupakan salah satu langkah penting sebelum mengimplementasikan kontroler pada sistem yang sesungguhnya. Dengan hasil simulasi sesuai dengan performa yang diharapkan, pengimplementasian kontroler pada sistem akan menjadi lebih mudah dan mencapai kriteria atau performa yang diinginkan.

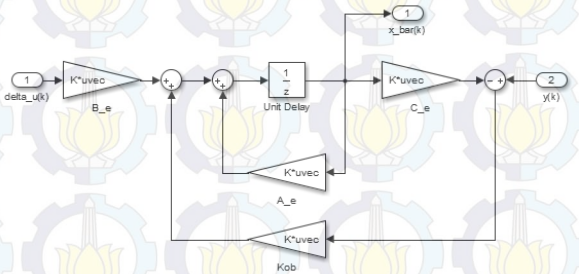
4.3.1 Diagram Blok Simulasi Sistem

Pengerjaan simulasi pada pelaksanaan Tugas Akhir kali ini akan dibantu oleh program MATLAB dan Simulink. Secara garis besar, diagram blok simulasi sistem terdiri dari *state-space* sistem, penguatan atau *gain* MPC dan subsistem *observer*. Blok diagram dari simulasi sistem dapat dilihat pada Gambar 4.2.

Seperti terlihat pada Gambar 4.2, terdapat blok diagram untuk *state-space* sistem yang terdiri dari blok *gain* A_p, B_p dan C_p. Selain itu terdapat blok penguatan pada kontroler MPC, yaitu K_y dan K_{MPC}. beserta sebuah blok *integrator* diskrit. Selain itu, ditambahkan blok *signal builder* sebagai penghasil nilai referensi atau *setpoint* pada sistem.Selain sistem diatas, ditambahkan pula satu subsistem berupa *observer*. Subsistem *observer* digunakan sebagai *state estimator* untuk mengestimasi suatu nilai *state* yang tidak diketahui. Subsistem *observer* dapat dilihat pada Gambar 4.3.



Gambar 4.2 Diagram Blok Simulasi Pengujian Kontroler MPC pada Sistem.



Gambar 4.3 Subsistem *Observer* pada Simulasi Sistem

4.3.2 Prosedur Pengerjaan Simulasi Sistem

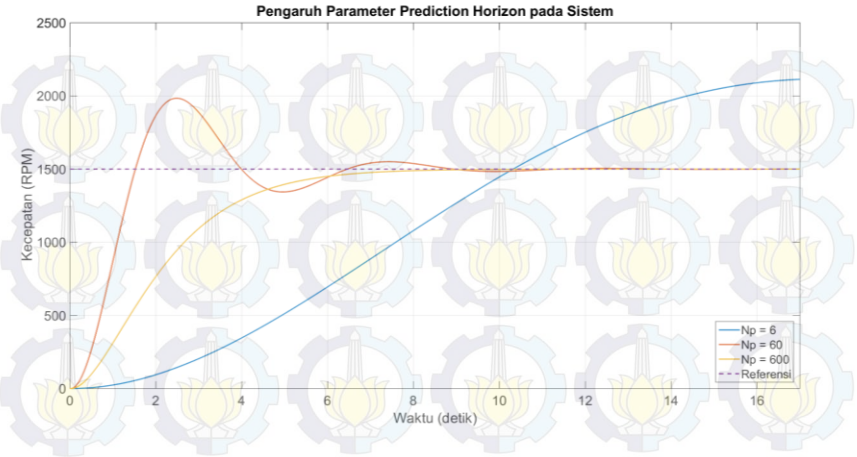
Pada langkah pengerjaan simulasi sistem, ada beberapa nilai dan variabel yang diamati pengaruhnya pada sistem. Variabel tersebut adalah nilai *prediction horizon* (N_p), *control horizon* (N_c) dan nilai *tuning parameter* pada indeks performansi (r_w). Ketiga variabel tersebut akan diuji satu persatu mengenai seberapa besar pengaruh ketiga variabel tersebut pada sistem. Untuk pengujian, sistem akan diberikan 2 input yang berbeda, yaitu *input unit step* dan *input tracking* sistem. Pada *tracking* sistem, nilai *setpoint* akan berubah-ubah dengan mengikuti *input ramp* pada kondisi kecepatan berbeda dan mengamati performa kontroler MPC dalam mengikuti dari nilai *setpoint* tersebut.

Setelah respon dari sistem yang diberi kontroler terlihat, maka langkah selanjutnya adalah menentukan karakteristik respon dari sistem tersebut. Parameter pada *input step* yang akan diamati adalah nilai *steady-state error*, *overshoot*, *settling time* dan *rise time*. Nilai *rise time* yang digunakan saat respon menyentuh angka 5% hingga 95%. Sedangkan *settling time* yang akan dihitung berada pada saat 2% dari nilai *steady state*. Sedangkan pada *input tracking* akan dilihat performa kontroler berdasarkan nilai *error steady-state* sistem.

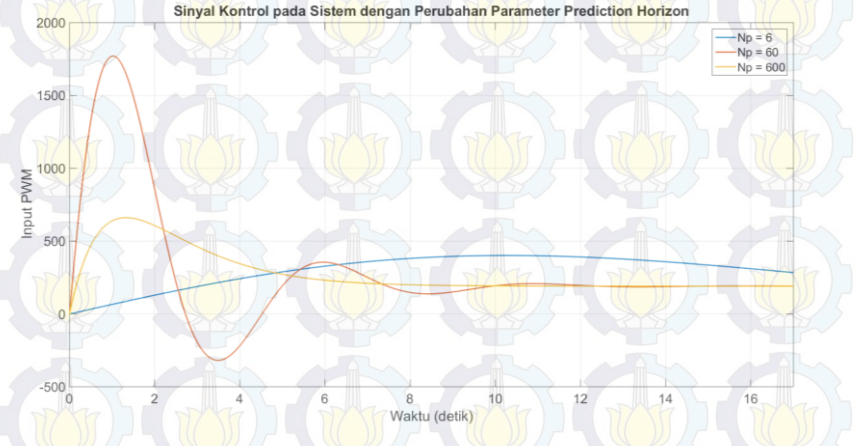
4.4 Pengaruh Parameter N_p pada Sistem

Variabel pertama yang akan diuji pada sistem dan kontroler MPC adalah nilai *prediction horizon* atau N_p. Nilai *prediction horizon* yang dijadikan parameter mempunyai nilai yang bervariasi, mulai dari satuan hingga ratusan. Setelah itu, akan dilihat karakteristik dari respon simulasi untuk menentukan nilai *prediction horizon* yang akan digunakan pada saat implementasi.

Terlihat dari Gambar 4.4, bahwa respon yang dihasilkan oleh sistem akan berbeda-beda seiring dengan parameter *prediction horizon* yang berbeda pula. Sedangkan sinyal kontrol yang dihasilkan oleh kontroler MPC dapat dilihat pada Gambar 4.5. Pada pengujian *prediction horizon* ini, nilai *control horizon* dan *tuning parameter* akan dijadikan nilai tetap, untuk melihat pengaruh dari perubahan nilai *prediction horizon* pada sistem. Pada pembebanan minimal dan nilai N_p = 6, respon yang dihasilkan sama sekali tidak mengikuti respon *unit step*. Respon terlihat baru mengikuti *tracking setpoint* yang diberikan ketika nilai *prediction horizon* bernilai dua digit ke atas, dalam hal ini diambil nilai 60 dan 600.



Gambar 4.4 Pengaruh Parameter *Prediction Horizon* pada Sistem.



Gambar 4.5 Sinyal Kontrol pada Sistem dengan Perubahan Parameter *Prediction Horizon*.

Pada nilai N_p = 60, terlihat bahwa respon sudah mengikuti *unit step* yang diberikan. Tetapi, seperti terlihat pada Gambar 4.4, *overshoot* yang dihasilkan oleh sistem sangat tinggi. Tentu saja respon seperti ini sangat dihindari karena akan merusak motor. Oleh karena itu, perlu di-*tuning* lagi

nilai dari *prediction horizon* agar respon yang dih.asilkan tidak mengalami *overshoot*.

Setelah melakukan beberapa kali pengujian, akhirnya ditemukan nilai N_p = 600. Respon yang dihasilkan jauh lebih baik dibandingkan nilai N_p = 60. Respon yang dihasilkan tidak mengalami *overshoot* sama sekali. Akan tetapi, terlihat bahwa respon yang dihasilkan lebih lambat daripada nilai N_p = 60. Untuk melihat karakteristik respon sistem dari tiap pembebanan, dapat dilihat pada subbab dibawah ini.

4.4.1 Beban Minimal

Pada pembebanan minimal, nilai *prediction horizon* yang diuji berada pada nilai 5, 50 dan 500. Untuk 2 parameter lainnya, seperti *control horizon* dan *tuning parameter* akan diberi nilai tetap dengan N_c = 2 dan r_w = 10. Tabel 4.2 menyajikan data mengenai nilai karakteristik respon untuk tiap-tiap nilai parameter *predicion horizon*.

Tabel 4.2 Karakteristik Respon Sistem pada Pembebanan Minimal untuk Tiap-Tiap Variabel *Prediction Horizon*.

N_p	*Steady-State Error (%)*	*Rise Time* (Detik)	*Settling Time* (Detik)	Presentase *Overshoot (%)*
5	Tidak mengikuti referensi			
50	0	1,25	6,76	33,26
500	0	7,66	10,34	0

Terlihat pada Tabel 4.2, respon sistem dengan nilai N_p = 5 tidak mengikuti referensi *unit step*. Sedangkan nilai *prediction horizon* sebesar 50 memiliki respon yang lebih cepat dibandingkan dengan sistem yang mempunyai nilai *prediction horizon* 500. Akan tetapi, respon yang cepat ini memiliki kelemahan yang sangat menonjol, yaitu nilai *overshoot*-nya yang tinggi.

4.4.2 Beban Nominal

Prosedur pengujian respon sistem pada beban nominal mempunyai langkah yang sama seperti pada pembebanan minimal. Hanya saja, nilai *prediction horizon* yang diuji pada sistem mempunyai nilai 4, 40 dan 400. Sedangkan nilai *control horizon* dan *tuning parameter* dibuat tetap pada nilai N_c = 3 dan r_w = 20. Karakteristik respon pada pembebanan nominal dapat dilihat pada Tabel 4.3.

Tabel 4.3 Karakteristik Respon Sistem pada Pembebanan Nominal untuk Tiap-Tiap Variabel *Prediction Horizon*.

N_p	*Steady-State Error (%)*	*Rise Time* (Detik)	*Settling Time* (Detik)	Presentase *Overshoot (%)*
4	Tidak mengikuti referensi			
40	0	1,62	16,77	52,57
400	0	5,70	7,74	0

Respon sistem pada pembebanan nominal mempunyai kemiripan respon yang sama seperti pembebanan minimal jika nilai *prediction horizon*-nya diubah-ubah. Terlihat pada Tabel 4.3, respon sistem dengan nilai $N_p = 4$ tidak mengikuti referensi *unit step*. Sedangkan nilai *prediction horizon* sebesar 40 memiliki respon yang lebih baik dibandingkan *prediciton horizon* 4 walaupun terdapat nilai *overshoot* yang lebih tinggi. Untuk mengkompensasi nilai *overshoot* yang tinggi ini, maka nilai N_p dinaikkan hingga mencapai 400. Respon yang dihasilkan tidak mempunyai *overshoot* sama sekali, walaupun nilai *rise time* sistem menjadi lebih besar. Artinya, respon akan menjadi lebih lambat.

4.4.3 Beban Maksimal

Pada pengujian nilai *prediction horizon* terakhir, akan dilakukan pada pembebanan maksimal. Nilai *prediction horizon* yang digunakan pada sistem mempunyai nilai 6, 60 dan 600. Sedangkan nilai *control horizon* dan *tuning parameter* dibuat tetap pada nilai $N_c = 4$ dan $r_w = 30$. Karakteristik respon pada pembebanan nominal dapat dilihat pada Tabel 4.4 berikut ini.

Pada, Tabel 4.4, dapat diamati bahwa respon yang dihasilkan pada tiap-tiap nilai *prediction horizon* mempunyai karakteristik yang serupa seperti pada pembebanan minimal dan nominal. Pada variabel *prediction*

Tabel 4.4 Karakteristik Respon Sistem pada Pembebanan Maksimal untuk Tiap-Tiap Variabel *Prediction Horizon*.

N_p	*Steady-State Error (%)*	*Rise Time* (Detik)	*Settling Time* (Detik)	Presentase *Overshoot (%)*
6	Tidak mengikuti referensi			
60	0	1,52	8,18	32,28
600	0	5,01	6,65	0

horizon yang paling rendah, respon masih tidak mengikuti referensi. Pada *prediction horizon* 30 dan 300 barulah respon sistem mengikuti referensi dari *unit step*. Respon dengan nilai *prediction horizon* 30 mempunyai respon yang lebih cepat walaupun nilai *overshoot* yang tinggi. Sedangkan nilai *prediction horizon* 300 mempunyai respon yang lebih lambat, tetapi tidak memiliki nilai *overshoot*.

4.5 Pengaruh Parameter N_c pada Sistem

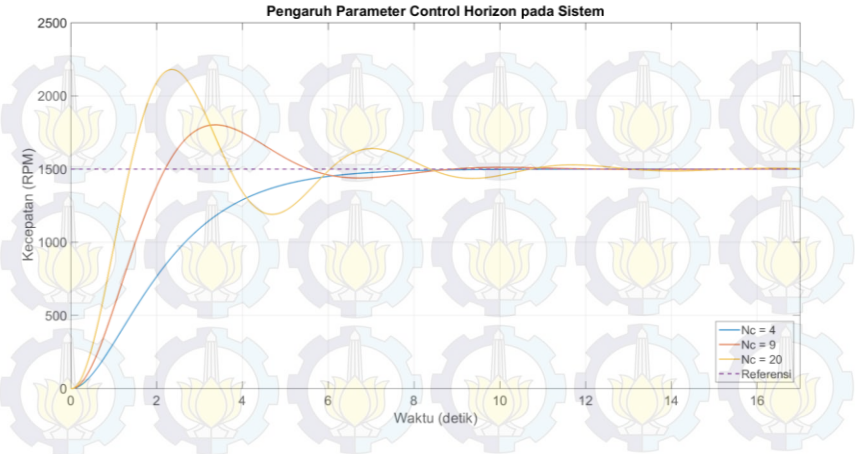
Setelah menentukan pengaruh dari *prediction horizon* pada srespon sistem, variabel selanjutnya yang akan diuji adalah pengaruh dari *control horizon* atau N_c pada respon sistem. Sama seperti *prediction horizon*, nilai dari *control horizon* yang diuji bervariasi dari satuan hingga ratusan. Nilai *control horizon* yang paling baik respon sistemnya akan dimasukkan sebagai parameter pada implementasi. Gambar 4.6 menunjukkan pengaruh dari perubahan nilai *control horizon* pada respon sistem.

Pada pengujian control *horizon* ini, nilai *prediction horizon* dan *tuning parameter* akan dijadikan nilai tetap, untuk melihat pengaruh dari perubahan nilai *control horizon* pada sistem. Terlihat pada gambar bahwa semua nilai variabel *control horizon* mengikuti dari *unit step*. Hanya saja, terlihat bahwa semakin kecil nilai dari *control horizon*, maka respon akan semakin lambat dan respon yang dihasilkan mirip seperti orde 1. Sebaliknya, semakin besar nilai *control horizon*, maka respon dari sistem akan semakin cepat. Akan tetapi, nilai *control horizon* yang semakin tinggi akan mengakibatkan semakin tinggi pula *overshoot* yang dihasilkan. Sinyal kontrol yang dihasilkan akibat perubahan dari parameter *control horizon* dapat dilihat pada Gambar 4.7. Untuk melihat pengaruh dari *control horizon* pada setiap pembebanan, dapat dilihat pada subbab di bawah ini.

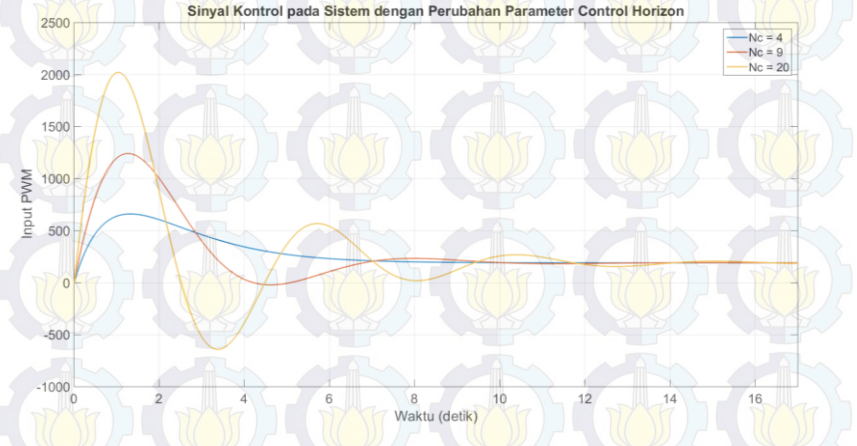
4.5.1 Beban Minimal

Pada pembebanan minimal, nilai *control horizon* yang diuji berada pada nilai 2, 10 dan 25. Untuk 2 parameter lainnya, seperti *prediction horizon* dan *tuning parameter* akan diberi nilai tetap dengan N_p = 500 dan r_w = 10. Tabel 4.5 menyajikan data mengenai nilai karakteristik respon untuk tiap-tiap nilai parameter *control horizon*.

Terlihat pada Tabel 4.5, semakin kecil nilai dari *control horizon*, semakin baik pula respon yang dihasilkan dilihat dari nilai *overshoot*.



Gambar 4.6 Pengaruh *Control Horizon* pada Sistem.



Gambar 4.7 Sinyal Kontrol pada Sistem dengan Perubahan Parameter *Control Horizon*.

Akan tetapi, akibat dari semakin kecilnya nilai *overshoot*, waktu respon yang dihasilkan menjadi semakin lambat. Akan tetapi, dipilih nilai *control horizon* yang paling kecil dikarenakan respon dengan *control horizon* paling kecil tidak menghasilkan *overshoot* sama sekali. Tentu

saja respon ini dibutuhkan karena respon dengan *overshoot* yang tinggi tentu dihasilkan dari sinyal kontrol yang tinggi juga, yang dapat berujung pada rusaknya motor BLDC.

Tabel 4.5 Karakteristik Respon Sistem pada Pembebanan Minimal untuk Tiap-Tiap Variabel *Control Horizon*.

N_c	*Steady-State Error (%)*	*Rise Time (Detik)*	*Settling Time* (Detik)	Presentase *Overshoot (%)*
2	0	7,66	10,37	0
10	0	1,53	8,32	33,26
25	0	1,12	8,07	49,80

4.5.2 Beban Nominal

Setelah mengetahui respon dari pembebanan minimal, langkah selanjutnya adalah menguji pengaruh *control horizon* pada pembebanan nominal. Nilai *control horizon* yang akan dimasukkan berada pada nilai 5, 15 dan 30. Sedangkan nilai *prediction horizon* dan *tuning parameter* akan dibuat konstan dengan N_p = 400 dan r_w = 20. Karakteristik respon pada sistem dengan pembebanan nominal dapat dilihat pada Tabel 4.6 berikut ini.

Terlihat pada Tabel 4.6, semakin kecil nilai dari *control horizon*, semakin baik pula respon yang dihasilkan dilihat dari nilai *overshoot*. Ini terlihat pada nilai *control horizon* paling rendah, respon yang diberikan menyerupai orde 1 dan tidak terdapat *overshoot*. Akan tetapi, nilai *rise time* pada *control horizon* bernilai 3 mempunyai nilai yang lebih besar dan berakibat pada respon yang lambat. Respon lambat ini bisa dibuat semakin cepat dengan meningkatkan nilai *control horizon*. Akan tetapi, peningkatan respon dari sistem bakat berakibat pada semakin tinggi pula nilai dari *overshoot* sistem.

Tabel 4.6 Karakteristik Respon Sistem pada Pembebanan Nominal untuk Tiap-Tiap Variabel *Control Horizon*.

N_c	*Steady-State Error (%)*	*Rise Time (Detik)*	*Settling Time* (Detik)	Presentase *Overshoot (%)*
3	0	5,70	7,74	0
15	0	1,74	11,64	37,13
30	0	1,43	12,87	46,67

Untuk implementasi pada pembebanan nominal, akan digunakan parameter *control horizon* sebesar 3. Ini dikarenakan respon tidak mengalami *overshoot* yang akan mengakibatkan gangguan pada sistem.

4.5.3 Beban Maksimal

Pengujian *control horizon* terakhir adalah pengujian pada sistem dengan pembebanan maksimal. Nilai *control horizon* yang digunakan berada pada nilai 4, 9 dan 20. Nilai *prediction horizon* dan *tuning parameter* akan dibuat konstan dengan $N_p = 600$ dan $r_w = 30$. Karakteristik respon pada sistem dengan pembebanan nominal dapat dilihat pada Tabel 4.7 berikut ini.

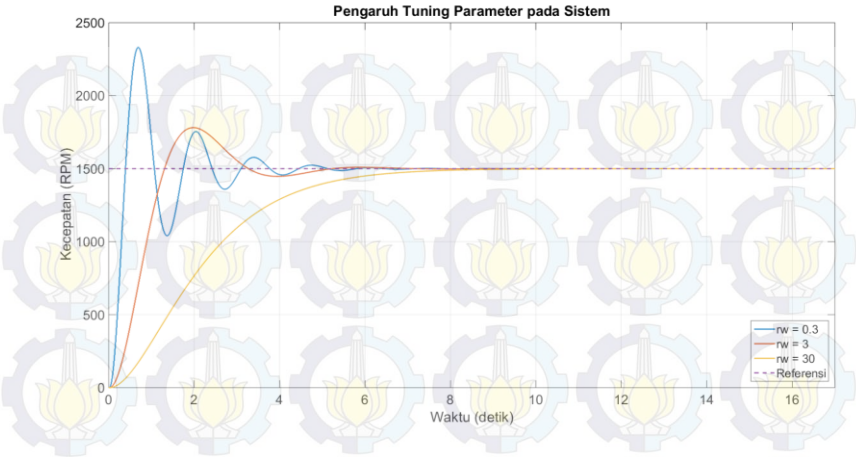
Tabel 4.7 Karakteristik Respon Sistem pada Pembebanan Maksimal untuk Tiap-Tiap Variabel *Control Horizon*.

N_c	*Steady-State Error (%)*	*Rise Time (Detik)*	*Settling Time* (Detik)	Presentase *Overshoot (%)*
4	0	5,01	6,65	0
9	0	2,19	7,93	20,15
20	0	1,35	11,69	45,47

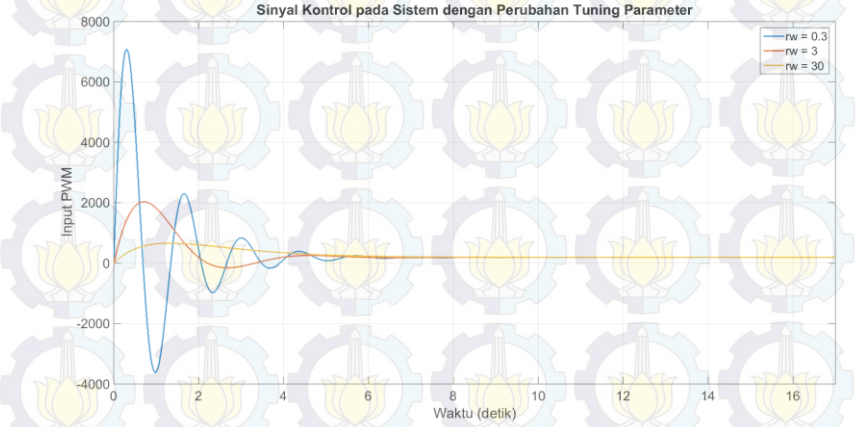
Tabel 4.7 menunjukkan, bahwa respon sistem yang paling baik dihasilkan oleh sistem dengan nilai *control horizon* sebesar 4. Respon dikatakan baik karena respon sistem yang dihasilkan tidak memiliki *overshoot* walaupun respon sistem lebih lambat. Cara untuk menaikkan waktu respon sistem dapat ditempuh dengan menaikkan nilai *control horizon*. Akan tetapi, perlu berhati-hati dalam meningkatkan waktu respon dari sistem. Semakin cepat respon, maka semakin tinggi pula *overshoot* yang dihasilkan. Ini terlihat dari nilai *control horizon* 9 dan 20 yang menunjukkan semakin rendah *rise time* yang dihasilkan, semakin tinggi pula *overshoot* yang dihasilkan.

4.6 Pengaruh Parameter r_w pada Sistem

Variabel terakhir yang diuji pengaruhnya pada sistem adalah nilai *tuning parameter* r_w pada indeks performansi *J*. Nilai dari *tuning parameter* ini mempunyai kemiripan dengan matriks pembobot *Q* dan *R*. pada kontroler LQR. Sama seperti prosedur pengujian sebelumnya, nilai dari *prediction horizon* dan *control horizon* akan dibuat konstan untuk mengetahui pengaruh dari *tuning parameter* pada indeks performansi dan sinyal kontrolnya, seperti terlihat pada Gambar 4.8 dan Gambar 4.9.



Gambar 4.8 Pengaruh Parameter *Tuning* Indeks Performansi pada Sistem



Gambar 4.9 Sinyal Kontrol pada Sistem dengan Perubahan *Tuning* Parameter pada Indeks Performansi.

Gambar 4.8 menunjukkan pengaruh dari perubahan nilai *tuning parameter* pada respon sistem. Terlihat pada gambar bahwa semua nilai variabel *tuning parameter* mengikuti dari *unit step*. Terlihat bahwa

semakin kecil nilai r_w, semakin cepat respon yang didapatkan. Sebaliknya, semakin besar nilai r_w, maka respon dari sistem akan semakin lambat. Akan tetapi, nilai *tuning parameter* yang semakin kecil akan mengakibatkan meningkatnya *overshoot* yang dihasilkan.

4.6.1 Beban Minimal

Pengujian *tuning parameter* pertama ada pada kondisi pembebanan minimal. Nilai *tuning parameter* yang digunakan berada pada nilai 0,1; 1 dan 10. Nilai *prediction horizon* dan *control horizon* akan dibuat konstan dengan N_p = 500 dan N_c = 2. Karakteristik respon pada sistem dengan pembebanan nominal dapat dilihat pada Tabel 4.8 berikut ini.

Tabel 4.8 Karakteristik Respon Sistem pada Pembebanan Minimal untuk Tiap-Tiap Variabel *Tuning Parameter*.

r_w	*Steady-State Error (%)*	*Rise Time* (Detik)	*Settling Time* (Detik)	Presentase *Overshoot (%)*
0.1	0	0,58	3,07	30
1	0	2,69	3,52	0
10	0	7,63	10,4	0

Tabel 4.8 menunjukkan bahwa semakin kecil nilai *tuning parameter*, semakin cepat pula respon dari sistem. Ini diperlihatkan dari nilai *rise time* yang semakin kecil. Akan tetapi, nilai *tuning parameter* yang semakin kecil akan mengakibatkan munculnya *overshoot* pada sistem. Untuk mengurangi dan meminimalkan *overshoot* sistem, maka nilai dari *tuning parameter* ini dinaikkan sedemikian sehingga *overshoot* semakin berkurang. Terlihat pada nilai r_w yang tinggi, pada nilai 1 dan 10, tidak terdapat *overshoot* pada sistem. Akan tetapi, apabila dibandingkan nilai *rise time* pada nilai r_w sama dengan 1 dan 10, respon akan semakin cepat apabila nilai *tuning parameter* semakin kecil. Oleh karena itu, dipilih nilai r_w sebesar 1 dengan pertimbangan tidak adanya respon *overshoot* dan *rise time* yang cepat.

4.6.2 Beban Nominal

Pengujian selanjutnya adalah melihat pengaruh *tuning parameter* pada pembebanan nominal. Rentang nilai *tuning parameter* yang digunakan pada pengujian berada pada nilai 0,2; 2, dan 20. Karakteristik respon dapat dilihat pada Tabel 4.9.

Tabel 4.9 Karakteristik Respon Sistem pada Pembebanan Nominal untuk Tiap-Tiap Variabel *Tuning Parameter*.

r_w	*Steady-State Error (%)*	*Rise Time* (Detik)	*Settling Time* (Detik)	Presentase *Overshoot (%)*
0,2	0	0,56	3,85	38,7
2	0	2,48	4,11	3,40
20	0	5,70	7,74	0

Terlihat pada Tabel 4.9 bahwa perubahan nilai *tuning parameter* mempunyai pengaruh yang cukup signifikan. Semakin kecil nilai *tuning parameter*, semakin cepat respon yang dihasilkan sistem walaupun nilai *overshoot* yang semakin tinggi. Untuk mengkompensasi nilai *overshoot* yang tinggi, maka nilai *tuning parameter* perlahan dinaikkan walaupun respon sistem semakin lambat.

4.6.3 Beban Maksimal

Pengujian *tuning parameter* terakhir adalah pengujian pada sistem dengan pembebanan maksimal. Nilai *tuning parameter* yang digunakan berada pada nilai 0,3; 3 dan 30. Sedangkan nilai *prediction horizon* dan *control horizon* akan dibuat konstan dengan N_p = 600 dan N_c = 4. Karakteristik respon pada sistem dengan pembebanan maksimal dapat dilihat pada Tabel 4.10 berikut ini.

Tabel 4.10 menunjukkan bahwa nilai *tuning parameter* yang semakin besar akan meningkatkan waktu respon dari sistem. Ini terlihat dari nilai *rise time* yang semakin kecil pada nilai *tuning parameter* yang semakin kecil. Akan tetapi, penggunaan nilai r_w yang telalui kecil sebaiknya di hindari dikarenakan adanya *overshoot* yang dihasilkan sistem. Nilai *overshoot* ini mungkin terlihat semakin besar, ketika nilai r_w semakin kecil.

Tabel 4.10 Karakteristik Respon Sistem pada Pembebanan Maksimal untuk Tiap-Tiap Variabel *Tuning Parameter*.

r_w	*Steady-State Error (%)*	*Rise Time* (Detik)	*Settling Time* (Detik)	Presentase *Overshoot (%)*
0,3	0	0,39	4,25	55,35
3	0	1,30	4,59	18,77
30	0	5,01	6,65	0

Oleh karena itu, alternatif lain ada pada penggunaan nilai r_w = 30. Hal ini dikarenakan nilai *tuning parameter* mempunyai respon yang menyerupai orde 1 dan tidak adanya *overshoot*. Walaupun begitu, nilai *rise time* yang dihasilkan semakin lambat. Oleh karena itu, untuk pengujian implementasi digunakan nilai *tuning parameter* sebesar 30.

4.7 Respon *Tracking* Sistem

Setelah mengetahui respon sistem setelah diberikan masukan *unit step*, analisa selanjutnya adalah mengetahui respon sistem saat diberikan *input* berupa referensi *tracking*. Referensi *tracking* yang diberikan meliputi sinyal *uji ramp* menuju kecepatan tertentu. Untuk keperluan analisa, akan diuji berapa besar pengaruh *prediction horizon, control horizon* dan *tuning parameter* ketika sistem diberi referensi *tracking* pada setiap pembebanan. Untuk analisa dari respon *tracking* sistem, akan diuji seberapa besar nilai *error steady-state* pada *unit ramp*, nilai RMSE (*Root Mean Square Error*), dan *overshoot* dari respon *tracking* yang dihasilkan.

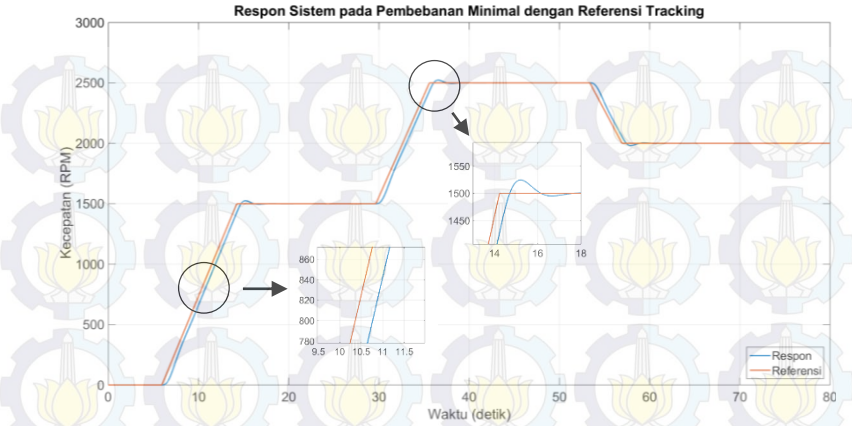
4.7.1 Beban Minimal

Pada pembebanan minimal, parameter kontroler MPC yang akan diberikan pada sistem bernilai N_p = 50; N_c = 2 dan r_w = 0,1. Setelah hasil didapatkan, barulah respon dianalisa karakteristik responnya seperti terlihat pada Gambar 4.10.

Terlihat pada Gambar 4.10, respon yang dihasilkan sudah dapat mengikuti referensi *tracking* yang diberikan. Hanya saja, terlihat bahwa terdapat nilai perbedaan antara respon dan referensi saat sistem diberikan *input unit ramp*. Selain itu, terdapat pula nilai *overshoot* ketika sistem telah memasuki nilai konstan pada kecepatan 1.500, 2.000 dan 2.500 RPM. Akan tetapi, nilai *error* tersebut sangatlah kecil sehingga dapat diabaikan. Untuk lebih jelasnya, karakteristik respon saat diberi referensi *tracking* dapat dilihat pada Tabel 4.11.

Tabel 4.11 Karakteristik Respon Sistem pada Pembebanan Minimal dengan Referensi *Tracking*.

Parameter	Nilai
Steady-State Error (%)	0,057
Presentase *Overshoot* (%)	0,017
RMSE	0,096



Gambar 4.10 Respon Sistem pada Pembebanan Minimal dengan Referensi *Tracking*.

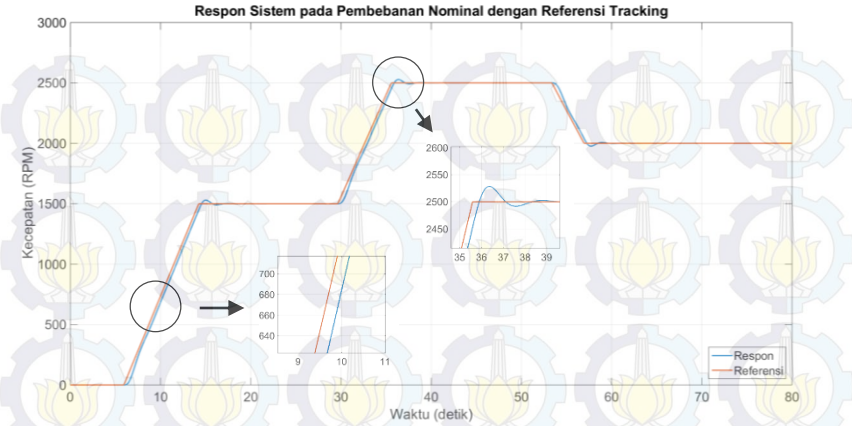
4.7.2 Beban Nominal

Setelah mendapatkan respon *tracking* pada pembebanan minimal, analisa selanjutnya adalah mengamati respon sistem pada pembebanan nominal jika sistem diberi referensi *tracking*. Pada pembebanan nominal, akan diberikan parameter kontroler MPC yang berbeda dengan pembebanan minimal. Parameter tersebut berupa N_p = 40; N_c = 3 dan r_w = 0,2. Hasil respon pembebanan nominal dapat dilihat pada Gambar 4.11.

Terlihat pada Gambar 4.11, respon yang dihasilkan sistem sudah mengikuti referensi *tracking* yang diberikan. Masalah yang sama seperti pada pembebanan minimal timbul pula pada respon di pembebanan nominal. Masih terdapat perbedaan nilai dan respon saat sistem mengikuti referensi *unit ramp* pada pada sistem. Selain itu, masih terdapat *overshoot* pada sistem ketika respon memasuki nilai referensi konstan. Karakteristik respon pada pembebanan no minal dapat dilihat pada Tabel 4.12.

Tabel 4.12 Karakteristik Respon Sistem pada Pembebanan Nominal dengan Referensi *Tracking*.

Parameter	Nilai
Steady-State Error (%)	0,041
Presentase *Overshoot* (%)	0,020
RMSE	0,092



Gambar 4.11 Respon Sistem pada Pembebanan Nominal dengan Referensi *Tracking*.

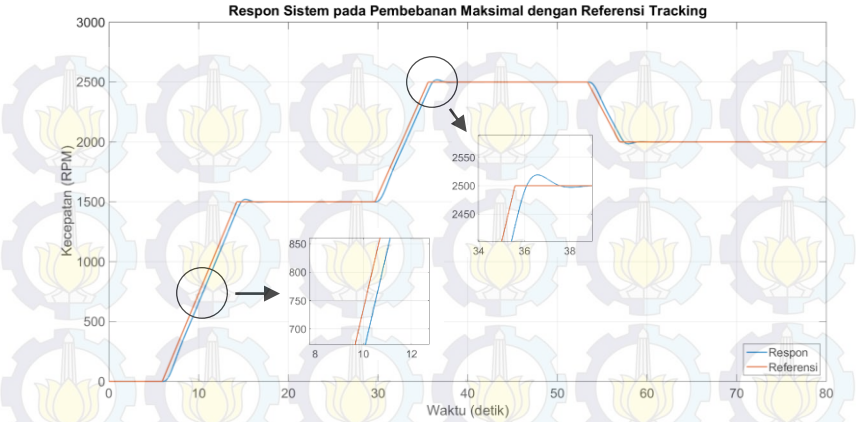
4.7.3 Beban Maksimal

Pengamatan terakhir untuk referensi tracking dilakukan pada pembebanan maksimal. Pada pembebanan maksimal, akan diberikan parameter kontroler MPC yang berbeda dengan pembebanan minimal maupun nominal. Parameter tersebut berupa $N_p = 60$; $N_c = 4$ dan $r_w = 0{,}3$. Hasil respon pada pembebanan maksimal dapat dilihat pada Gambar 4.12.

Respon yang terlihat pada Gambar 4.12 sudah dapat mengikuti referensi *tracking* yang diberikan. Selain itu, masih terdapat perbedaan respon ketika sistem mengikuti sinyal *unit ramp*. *Overshoot* pun masih terjadi ketika respon mulai memasuki nilai referensi konstan. Akan tetapi, nilai kedua parameter karakteristik respon tersebut sangatlah kecil hingga dapat dianggap tidak terlalu mempengaruhi performa dari motor BLDC ketika mengikuti referensi. Untuk lebih jelasnya, karakteristik respon saat diberi referensi *tracking* dapat dilihat pada Tabel 4.13.

Tabel 4.13 Karakteristik Respon Sistem pada Pembebanan Maksimal dengan Referensi *Tracking*.

Parameter	Nilai
Steady-State Error (%)	0,072
Presentase *Overshoot* (%)	0,013
RMSE	0,094



Gambar 4.12 Respon Sistem pada Pembebanan Maksimal dengan Referensi *Tracking*.

4.8 Implementasi Sistem

Setelah merancang dan menganalisa performa kontroler MPC pada tahapan simulasi, langkah selanjutnya adalah menerapkan dan mengimplementasikan kontroler pada sistem. Sebelumnya, akan dijelaskan telebih dahulu mengenai realisasi dari perancangan sistem yang telah dibuat pada Bab 3. Tahapan implementasi memiliki prosedur yang mirip dengan simulasi sistem. Hal yang berbeda terletak pada pergantian model matematika pada tahapan simulasi yang diganti dengan *plant* sesungguhnya melalui komunikasi *serial*. Untuk lebih jelasnya, blok diagram dan performa dari kontroler MPC pada tahapan sistem dapat dilihat pada subbab dibawah ini.

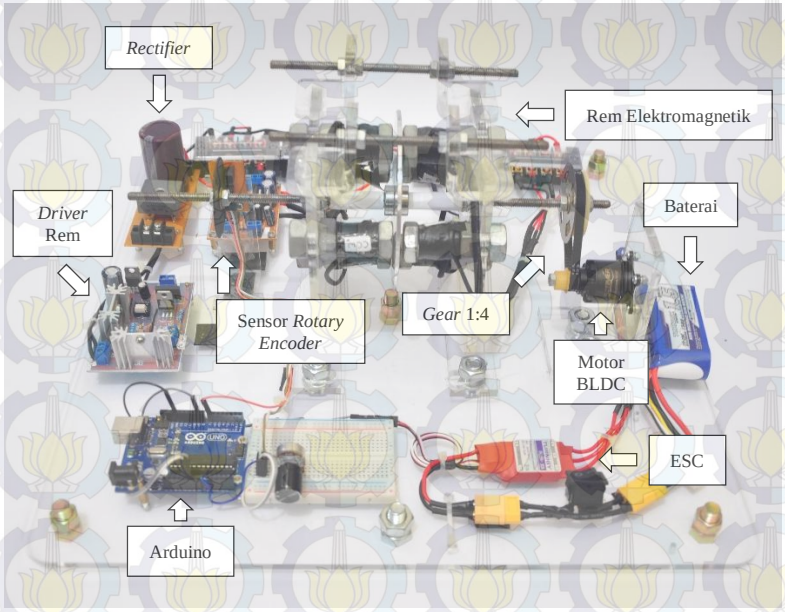
4.8.1 Realisasi Perancangan Sistem

Pada Bab 3, telah dijelaskan mengenai perancangan sistem untuk melaksanakan pengerjaan Tugas Akhir kali ini. Pada subbab ini, akan dijelaskan mengenai realisasi perancangan sistem keseluruhan, terutama bagian-bagian yang dikerjakan secara mandiri, seperti komponen sensor *rotary encoder* dan rem elektromagnetik. Selain itu, akan dijelaskan pula mengenai realisasi dan integrasi tiap-tiap komponen dari perancangan sistem secara keseluruhan.

Hal pertama yang akan dijelaskan terlebih dahulu adalah mengenai konstruksi sistem secara keseluruhan yang dapat dilihat pada Gambar

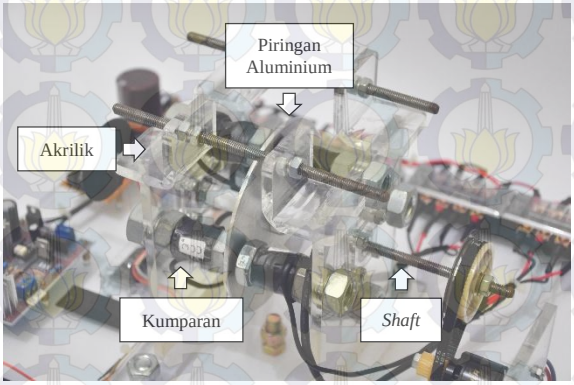
4.13. Pada gambar tersebut, terlihat bahwa motor BLDC dikopel dengan sistem *gear* 4:1 yang menghubungkan motor dengan *shaft*. Selain itu, motor BLDC tersebut disuplai tenaga dari baterai Lithium-Polimer dan dihubungkan ke ESC sebagai *driver* dari motor BLDC.

Sebagai salah satu komponen yang memiliki peranan yang sangat penting dalam memberi pembebanan pada motor, rem elektromagnetik dirancang untuk memberikan medan magnet yang kuat dengan suplai tegangan hingga 24 Volt. Rem tersebut tersusun dari kumparan lilitan tembaga yang mempunyai inti dari besi. Inti tersebut terbuat dari baut yang berdiameter 1,4 cm. Kumparan tersebut lalu disambung seri untuk tiap kutubnya dan dialiri tegangan yang diatur menggunakan metode PWM. Pengaturan tegangan yang masuk ke dalam kumparan ini diatur oleh driver dan Arduino. Sedangkan sumber tegangannya berasal dari listrik jala-jala PLN yang disearahkan dengan menggunakan komponen rectifier. Konstruksi rem elektromagnetik pada *plant* ditunjukkan seperti terlihat pada Gambar 4.14.

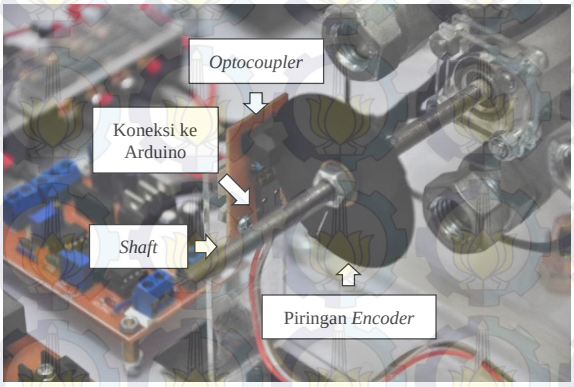


Gambar 4.13 Realisasi Perancangan Sistem Pengaturan Kecepatan Motor BLDC.

Selain rem elektromagnetik, komponen sensor *rotary encoder* juga mempunyai peranan yang penting untuk menghitung kecepatan dari motor BLDC. Sensor ini terdiri dari *optocoupler* dan resistor, serta diberi suplai tegangan sebesar 5 Volt. *Optocoupler* ini lalu dihubungkan dengan Arduino untuk menghitung banyaknya pulsa dalam satu detik, yang kemudian akan dikonversi ke dalam satuan kecepatan dengan algoritma tertentu. Hasil perancangan darisensor *rotary encoder* dapat dilihat pada Gambar 4.15.



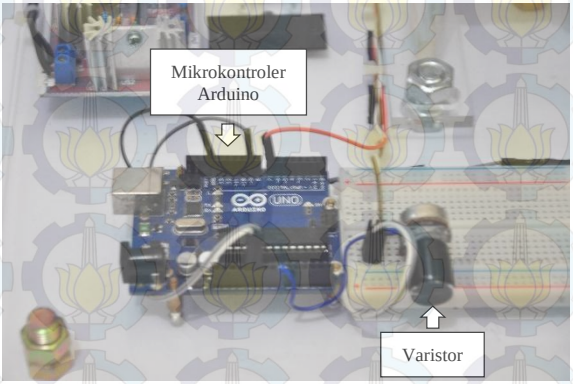
Gambar 4.14 Konstruksi Rem Elektromagnetik pada Sistem.



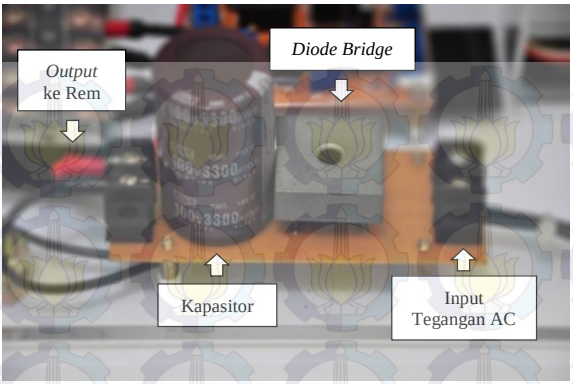
Gambar 4.15 Hasil Perancangan Sensor *Rotary Encoder*.

Mikrokontroler Arduino sebagai alat akuisisi data pada sistem ini juga mempunyai peranan yang sangat penting. Arduino merupakan tempat pengolahan dan algoritma penghitungan nilai kecepatan motor BLDC dilakukan. Selain itu, pada Arduino juga disambungkan sebuah *variable resistor*. *Variable resistor* (Varistor) ini berfungsi untuk mengubah masukan tegangan masukan pada rem elektromagnetik dari skala 0 sampai dengan 1000 melalui pin A0. Pada sisi lain, Arduino juga bertugas untuk memberi *output* PWM kepada ESC dan *driver* rem melalui pin 9 dan 11. Terakhir, *output* dari sensor *rotary encoder* disambungkan ke Arduino melalui pin 7. Tampilan mikrokontroler Arduino sebagai alat akuisisi data dapat dilihat pada Gambar 4.16.

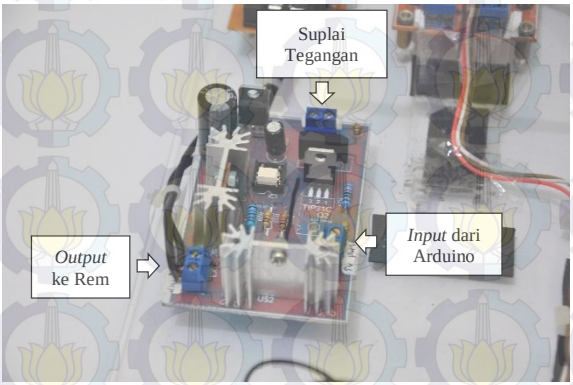
Seperti telah disebutkan di atas, tegangan yang masuk ke dalam rem elektromagnetik diatur oleh sebuah *driver* dan rangkaian *rectifier*. Pada rangkaian *rectifier*, sumber yang digunakan adalah jala-jala listrik PLN dan keluaran yang dihasilkan berupa tegangan searah yang disambungkan ke dalam rem elektromagnetik. Hasil dari rangkaian *rectifier* dapat dilihat pada Gambar 4.17. Rangkaian terakhir adalah rangkaian *driver* rem yang berfungsi mengatur tegangan masukan ke rem dalam bentuk PWM. Rangkaian ini diberi input tegangan DC sebesar 12 Volt. Sedangkan input untuk mengatur besarnya nilai PWM berasal dari mikrokontroler Arduino. Rangkaian *driver* rem dapat dilihat pada Gambar 4.18.



Gambar 4.16 Alat Akuisisi Data berupa Mikrokontroler Arduino.



Gambar 4.17 Rangkaian *Rectifier* pada Sistem



Gambar 4.18 Rangkaian *Driver* untuk Pengaturan Tegangan *Input* pada Rem Elektromagnetik.

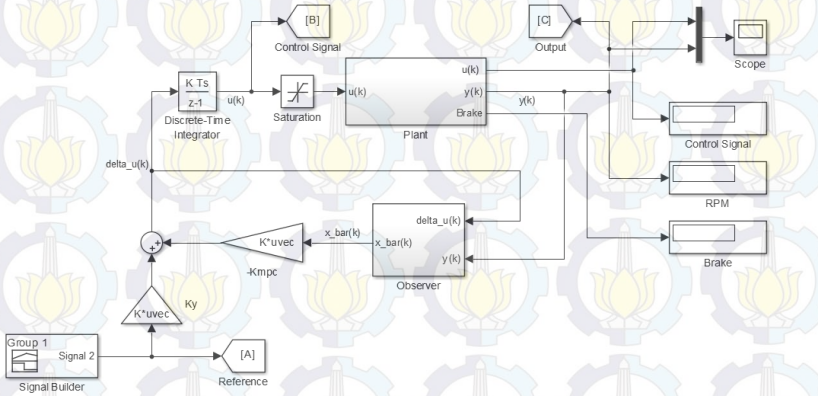
4.8.2 Diagram Blok Implementasi Sistem

Secara garis besar, perbedaan antara tahapan simulasi sistem dengan implementasi sistem hanya terletak pada pemodelan *plant*, khususnya motor BLDC. Jika pada tahap simulasi pemodelan *plant* digantikan dengan model matematika berbentuk *state-space*, tahap implementasi ini *output* kontroler langsung disalurkan ke dalam *driver* ESC pada *plant*. Untuk pembacaan nilai kecepatan motor BLDC dilakukan oleh sensor *rotary encoder* yang datanya dikirimkan kembali ke MATLAB melalui

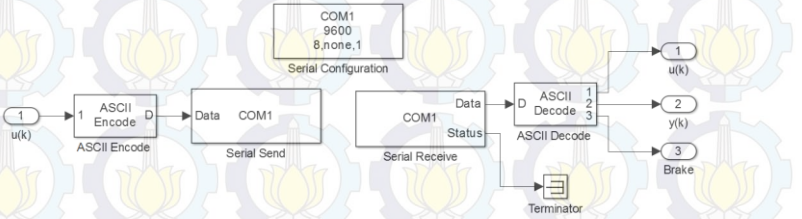
komunikasi *serial*. Selebihnya bentuk blok diagram implementasi sistem memilki kemiripan dengan tahapan simulasi sistem.

Pada Gambar 4.19, terlihat bahwa pemodelan matematika sistem dalam bentuk *state-space* telah digantikan dengan suatu subsistem bernama *plant*. Subsistem ini mempunyai 1 *input* berupa *u*(*k*) atau sinyal kontrol dari kontroler MPC. Sedangkan *output*-nya berjumlah 3 yang terdiri dari sinyal kontrol *u*(*k*), pembacaan nilai kecepatan dalam satuan RPM beserta nilai *input* pengereman (*brake*) yang diberikan pada sistem.

Isi dari subsistem *plant* yang merupakan komunikasi *serial* dapat dilihat pada Gambar 4.20 terlihat dari gambar bahwa blok komunikasi serial yang digunakan adalah *Serial Send*, *Serial Receive*, dan *Serial Configuration* yang terdapat pada *Instrument Control Toolbox*.



Gambar 4.19 Diagram Blok Implementasi Kontroler MPC pada Sistem.



Gambar 4.20 Subsistem *Plant* yang Terdiri Dari Komunikasi *Serial* pada Sistem.

Selain itu, digunakan blok *ASCII Decode* dan *ASCII Encode* yang berfungsi mengubah data menjadi menjadi bentuk ASCII yang dapat dibaca oleh Arduino.

4.8.3 Analisa Kontroler MPC pada Implementasi Sistem

Setelah menyiapkan diagram blok untuk tahap implementasi, langkah selanjutnya adalah menentukan nilai parameter kontroler MPC untuk tahapan implementasi sistem. Nilai parameter tersebut merupakan nilai *prediction horizon*, *control horizon* dan *tuning parameter* pada sistem. Nilai-nilai dari parameter tersebut didapat dari tahapan simulasi yang sebelumnya telah dilalui.

Berdasarkan tahapan simulasi, parameter dari kontroler MPC untuk tiap pembebanan dapat dilihat pada Tabel 4.14 berikut ini.

Setelah mengetahui nilai parameter kontroler MPC pada tiap-tiap pembebanan, langkah selanjutnya adalah menjalankan sistem yang sudah terpasang kontroler MPC dan melihat respon kecepatan yang diberikan pada tiap-tiap pembebanan dengan referensi *tracking*.

Terlihat semua pembebanan, pada respon yang diberikan oleh sistem dapat mengikuti referensi *setpoint* yang diberikan walaupun sistem, dalam hal ini motor BLDC, sedang diberikan pembebanan. Sistem dapat mengikuti referensi dengan baik pada kecepatan 1500 dan 2000 RPM. Akan tetapi, jika lebih diteliti secara seksama, terdapat *error steady-state* saat referensi berada pada kecepatan 2500 RPM. Respon yang diberikan juga terlihat tidak begitu halus mengikuti referensi *setpoint* yang diberikan.

Ada beberapa faktor yang menyebabkan respon terlihat tidak stabil di satu titik alias terdapat "riak" pada respon yang diberikan sistem. Faktor pertama berada pada sensor *rotary encoder* yang menerima banyak

Tabel 4.14 Nilai Parameter dari Kontroler MPC untuk Tiap–Tiap Nilai Pembebanan.

Parameter	Minimal	Nominal	Maksimal
Prediction Horizon N_p	50	40	60
Control Horizon N_c	2	3	4
Tuning Parameter r_w	0,1	0,2	0,3

noise dari lingkungan sekitar, contohnya adalah cahaya luar yang ikut masuk ke *optocoupler*, gangguan dari rem elektromagnetik yang disebabkan oleh kumparan rem yang jaraknya berdekatan dengan sensor, serta akurasi dari pembacaan dan kalibrasi pada sensor. Dikarenakan berbagai faktor diatas, pembacaan nilai kecepatan oleh sensor *rotary encoder* tidak bisa stabil di satu titik, melainkan naik dan turun.

Faktor kedua adalah pengestimasian nilai *state* lain yang tidak diketahui yang dilakukan oleh *observer* tidak dapat ditentukan secara akurat. Faktor ini disebabkan karena pembacaan sensor yang memang tidak stabil karena faktor pertama. Oleh sebab itu, *observer* tidak dapat mengestimasi nilai *state* lainnya secara akurat. Faktor terakhir adalah *data loss* yang terjadi pada komunikasi *serial*. *Data loss* terjadi apabila data *input* PWM yang dikirimkan MATLAB ke Arduino tiba-tiba terputus sehingga menganggu nilai *input* yang masuk ke *driver* ESC. Akibatnya, data *input* PWM yang dikirim ke *driver* ESC hilang untuk sepersekian detik sehingga kecepatan dari motor BLDC akan turun untuk sesaat. *Data loss* ini terjadi akibat *cache* dari mikrokontroler Arduino yang penuh, program *background* yang berjalan pada sistem operasi komputer yang dapat memperberat kinerja dari sistem dan faktor lainnya.

Subbab di bawah ini akan menjelaskan karakteristik respon untuk tiap pembebanan pada tahapan implementasi sistem.

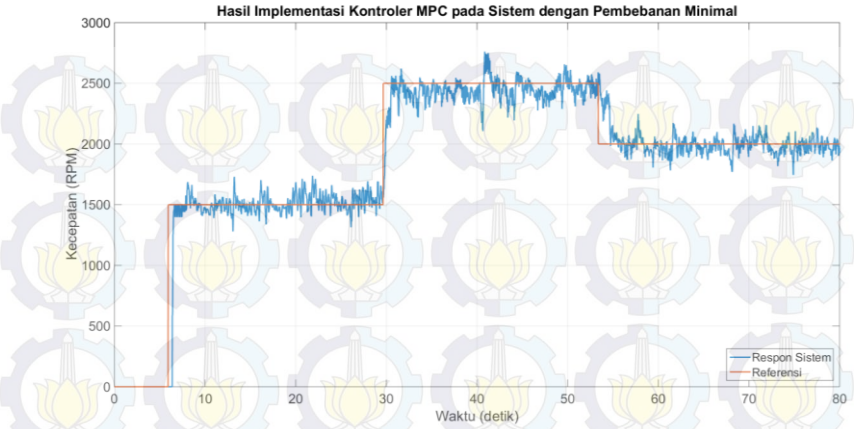
4.8.3.1 Beban Minimal

Pada pembebanan nominal, tegangan yang masuk ke dalam rem elektromagnetik sebesar 4,4 Volt untuk mengetahui apakah sistem dapat mengikuti referensi jika diberi beban berupa rem elektromagnetik. Sedangkan parameter kontroler MPC yang diberikan pada pembebanan nominal dapat dilihat pada Tabel 4.15.

Setelah tahap implementasi sistem pada pembebanan nominal, respon sistem lalu diamati untuk melihat karakterstiknya. Karakteristik respon untuk pembebanan minimal dapat dilihat pada Tabel 4.12.

Tabel 4.15 Karakteristik Respon dengan Pembebanan Minimal pada Implementasi Sistem.

Parameter	Nilai
Steady-State Error (%)	10
Rise Time (Detik)	0,78
Settling Time (Detik)	1,08
Presentase *Overshoot (%)*	0



Gambar 4.21 Hasil Implementasi Kontroler MPC pada Sistem dengan Pembebanan Minimal.

Pada Tabel 4.15, terlihat bahwa respon yang dihasilkan pada implementasi sistem mempunyai respon yang lebih cepat dibandingkan pada tahap simulasi. Akan tetapi, jika pada tahap simulasi tidak terdaapt *steady-state error* pada respon, hal ini berbeda dimana terdapat *steady-state error* sebesar 10%. Untuk tahap implementasi ini, tidak terdapat *overshoot* pada sistem, sama halnya seperti pada tahap simulasi. Respon *tracking* dan sinyal kontrol pada pembebanan minimal dapat dilihat pada Gambar 4.21.

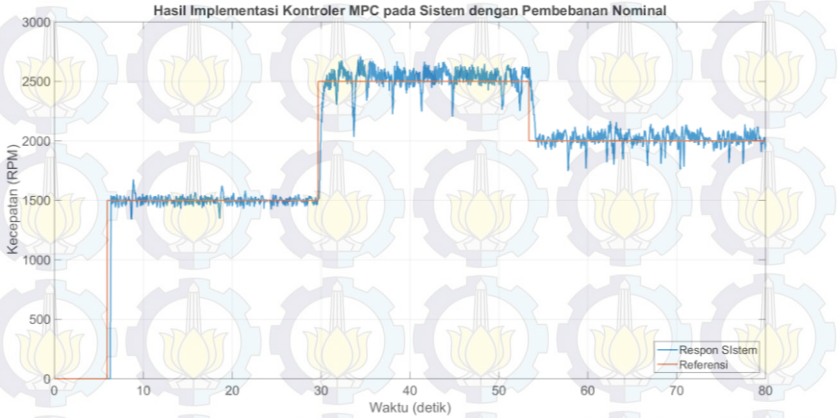
4.8.3.2 Beban Nominal

Setelah mengetahui respon sistem untuk pembebanan minimal, tahapan selanjutnya adalah mengetahui respon hasil implementasi untuk pembebanan nominal. Pada pembebanan nominal, sistem diberi beban berupa rem elektromagnetik dengan masukan tegangan sebesar 7,89 Volt. Sistem diberi referensi *tracking* dari kecepatan 1500 hingga 2000 RPM.

Setelah respon hasil implementasi didapat, langkah selanjutnya adalah mengamati karakteristik dari respon sistem saat pembebanan nominal. Karakteristik respon yang diamati adalah *settling time*, *rise time*, *error steady-state* dan *overshoot*. Karakteristik respon untuk pembebanan nominal dapat dilihat pada Tabel 4.16. Respon kecepatan dari motor

BLDC yang dihasilkan pada pembebanan nominal dapat dilihat pada Gambar 4.22.

Terlihat pada Tabel 4.16 bahwa masih terdapat nilai *steady-state error* pada pembebanan nominal, sama halnya seperti pada pembebanan minimal. Akan tetapi, nilai *steady-state error* pada pembebanan nominal mempunyai nilai lebih rendah dibandingkan pembebanan minimal. Untuk nilai *rise time* dan *settling time*, tahap implementasi mempunyai nilai yang lebih cepat dibandingkan tahap simulasi pada *software* MATLAB. Respon dari sistem lebih menyerupai orde 1 dikarenakan adanya nilai *overshoot*.



Gambar 4.22 Hasil Implementasi Kontroler MPC pada Sistem dengan Pembebanan Nominal.

Tabel 4.16. Karakteristik Respon dengan Pembebanan Nominal pada Implementasi Sistem.

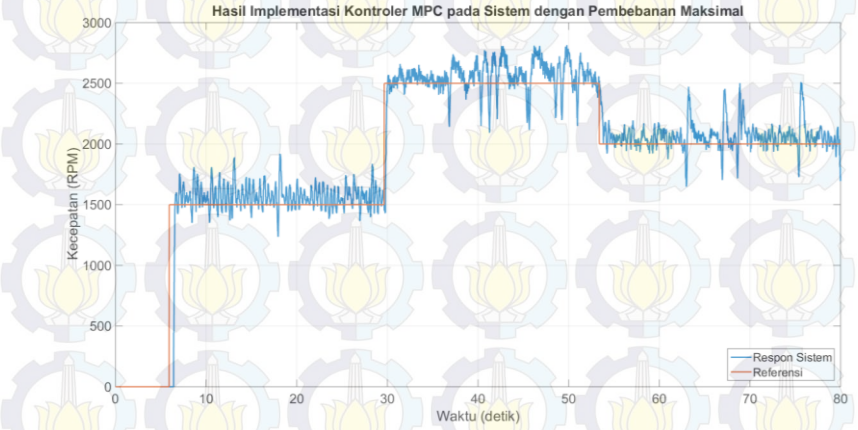
Parameter	Nilai
Steady-State Error (%)	7,4
Rise Time (Detik)	0,69
Settling Time (Detik)	1,27
Presentase *Overshoot (%)*	0

4.8.3.3 Beban Maksimal

Pada tahap implementasi terakhir, yaitu pada pembebanan maksimal, tegangan yang masuk ke dalam rem elektromagnetik sebesar 12,46 Volt untuk mengetahui apakah sistem dapat mengikuti referensi jika diberi beban berupa rem elektromagnetik. Referensi yang diberikan berupa *tracking* pada *setpoint* dengan nilai berkisar 1500 hingga 2500 RPM.

Setelah menyelesaikan tahap implementasi pada pembebanan maksimal, langkah selanjutnya adalah mengamati karakteristik respon dari sistem. Karakteristik respon yang diamati adalah settling *time*, *rise time*, *error steady-state* dan *overshoot*. Karakteristik respon untuk pembebanan maksimal dapat dilihat pada Tabel 4.17. Respon kecepatan dari motor BLDC yang dihasilkan pada pembebanan nominal dapat dilihat pada Gambar 4.23.

Tabel 4.17 menunjukkan bahwa masih terdapat *steady-state error* pada sistem, sebesar 8,9%. Nilai ini berbeda dibandingkan dengan tahapan simulasi dimana tidak adanya nilai *steady-stateg error* sama sekali. Sedangkan untuk kecepatan respon sistem, respon pada tahapan implementasi jauh lebih cepat dibandingkan pada simulasi. Ini dibuktikan dengan lebih kecilnya nilai *rise time* dan *settling time* pada tahap implementasi dibandingkan tahap simulasi.



Gambar 4.23 Hasil Implementasi Kontroler MPC pada Sistem dengan Pembebanan Maksimal.

Tabel 4.17 Karakteristik Respon dengan Pembebanan Maksimal pada Implementasi Sistem

Parameter	Nilai
Steady-State Error (%)	12
Rise Time (Detik)	0,94
Settling Time (Detik)	0.98
Presentase *Overshoot (%)*	0

DAFTAR PUSTAKA

[1] Xia, C.L., "*Permanent Magnet Brushless DC Motor Drives and Controls"*, John Wiley & Sons Singapore Pte. Ltd., Singapore, 2012.

[1] Elrosa, Ilmiyah., "Traction Control pada Parallel Hybrid Electric Vehicle dengan Metode Generalized Predictive Control", *Tugas Akhir*, Jurusan Teknik Elektro, Institut Teknologi Sepuluh Nopember, Surabaya, 2014

[2] Janardana, Gede B.A., "Desain dan Analisis Motor Axial Flux Brushless DC Berbasis 3D Finite Element Method Untuk Aplikasi Kendaraan Listrik", *Tugas Akhir*, Jurusan Teknik Elektro, Institut Teknologi Sepuluh Nopember, Surabaya, 2015

[3] _____, "*2-2-1 What is a Brushless DC Motor ?*", <URL: http://www.nidec.com/en-NA/technology/motor/basic/00018/>, Maret, 2015

[4] _____, "*Brushless DC Motor*", <URL: http://hades.mech.northwestern.edu/index.php/Brushless_DC_Motors>, Maret, 2014

[5] _____, "*How Hall-Effect Sensors Impact Motor Energy Use*", <URL: http://www.automationworld.com/sensors-discrete/how-hall-effect-sensors-impact-motor-energy-use>, Maret, 2015

[6] _____, "*Brushless Motor DC Animation*", <URL: http://www.avdweb.nl/solar-bike/hub-motor/permanent-magnet-dc-hub-motor-tuning.html>, Maret, 2015

[7] _____, "*How are Magnets Used in Amusement Park Rides?*", <URL: https://period7magnets.wikispaces.com/How+are+magnets+used+in+amusement+park+rides%3F>, April, 2015

[8] Permana, Yoki., "Perancangan dan Implementasi Pengaturan Kecepatan Motor 3 fasa pada Mesin Sentrifugal Menggunakan Metode Model Reference Adaptive Control (MRAC)", *Tugas Akhir*, Jurusan Teknik Elektro, Institut Teknologi Sepuluh Nopember, Surabaya, 2014

[9] _____, "*The MCU and Other Components of The NC System*", <URL: https://rekadayaupaya.wordpress.com/2013/06/13/8-4-the-mcu-and-other-components-of-the-nc-system/>, April, 2015

[10] Söderström, T. and Stoica, P., "*System Identification"*, Prentice Hall, United Kingdom, 2001

[11] Wang, L., "*Model Predicitive Control System Design and Implementation Using MATLAB*", Springer, London, 2009

[12] Seborg, D.E., Mellichamp, D.A., Edgar, T.F., dan Doyle, F.J., "*Process Dynamic and Control*", Wiley, London, 2011
[13] _____, "*HP2212-1000KV Outrunner Brushless Motor For Quadrotor*", <URL: http://rctimer.com/product-813.html>, April, 2015
[14] _____, "*TURNIGY Plush 30Amp Speed Controller*", <URL: http://www.hobbyking.com/hobbyking/store/__2164__TURNIGY_Plush_30amp_Speed_Controller.html>, April, 2015
[15] _____, "*SparkFun Low Current Sensor Breakout - ACS712*", <URL: https://www.sparkfun.com/products/8883>, April, 2015
[16] _____, "*Arduino*", <URL:http://en.wikipedia.org/wiki/Arduino>, April, 2015
[17] Ogata, Katsuhiko., "*Discrete-Time Control System*", Prentice Hall, New Jersey, 1994
[18] Ogata, Katsuhiko., "*Modern Control Engineering 4th Edition"*, Prentice Hall, New Jersey, 2002

BAB 5
PENUTUP

Bab 5 merupakan bab terakhir pada laporan Tugas Akhir ini yang akan menjelaskan tentang kesimpulan yang dapat ditarik setelah melakukan proses pengerjaan Tugas Akhir ini. Pada bab ini juga disertakan mengenai saran yang dapat dipertimbangkan dalam pengerjaan Tugas Akhir selanjutnya.

5.1 Kesimpulan

Berdasarkan hasil simulasi dan implementasi sistem pengaturan kecepatan motor BLDC menggunakan kontroler *Model Predictive Control* (MPC), dapat ditarik beberapa kesimpulan sebagai berikut:

a. Hasil simulasi dan implementasi kontroler MPC pada sistem pengaturan kecepatan motor BLDC menunjukkan bahwa kecepatan motor BLDC dapat mengikuti referensi *tracking* pada semua parameter pembebanan (minimal, nominal dan maksimal).
b. Penentuan nilai *prediction horizon* pada kontroler MPC sangat mempengaruhi respon dari sistem. Semakin besar nilai *prediction horizon*, respon yang dihasilkan akan semakin lambat dan halus. Sebaliknya, semakin kecil nilai *prediciton horizon* akan mengakibatkan respon yang semakin cepat. Akan tetapi, nilai *prediciton horizon* yang semakin kecil dapat menimbulkan *overshoot* yang semakin tinggi.
c. Selain *prediction horizon*, parameter *control horizon* pada kontroler MPC juga memilki pengaruh yang besar pada sistem. Jika nilai *control horizon* semakin besar, maka sistem akan memiliki waktu respon yang lebih cepat dengan kekurangan *overshoot* yang semakin tinggi. Adapun jika nilai *control horizon* semakin kecil, respon yang dihasilkan sistem akan semakin halus walaupun waktu respon akan semakin lambat.
d. Pada parameter terakhir, yaitu *tuning parameter* pada indeks performansi, juga mempunyai pengaruh yang signifikan pada sistem. Semakin besar nilai *tuning parameter*, respon yang dihasilkan akan semakin cepat walaupun terdapat kekurangan karena timbulnya *overshoot*. Sebaliknya, semakin kecil nilai

tuning parameter, respon yang dihasilkan akan semakin lambat dan menyerupai respon orde 1 tanpa adanya *overshoot*.

5.2 Saran

Setelah melalui banyak proses dan tahapan pada pengerjaan Tugas Akhir, penulis dapat memberikan berapa saran seperti berikut ini:

a. Terdapat beberapa parameter dari lingkungan luar yang dapat mempengaruhi kecepatan dari motor BLDC pada sistem atau *plant* ini. Antara lain kondisi permukaan pada *plant*, posisi *belt* yang menghubungkan motor dengan *shaft* dan beberapa parameter lainnya. Oleh karena itu, untuk pengerjaan Tugas Akhir selanjutnya dapat memperhatikan dan memperbaiki hal-hal minor diatas.
b. Pada pengerjaan Tugas Akhir ini, sumber tenaga yang digunakan untuk menjalankan motor BLDC adalah baterai *Lithium-Polimer* (Li-Po). Disarankan untuk pengerjaan Tugas Akhir selanjutnya, lebih mengutamakan penggunaan sumber tenaga listrik yang permanen (contohnya listrik PLN). Ini dikarenakan kapasitas dari baterai yang dapat mempengaruhi kecepatan dari motor BLDC.
c. Pada pengerjaan Tugas Akhir selanjutnya disarankan untuk mengganti sensor *rotary encoder* dengan menggunakan sensor *tachogenerator* ataupun sensor *rotary encoder* dengan standar industri. Ini untuk menjamin bahwa pembacaan kecepatan motor BLDC dapat berjalan secara presisi dan akurat.
d. Pada perancangan kontroler *Model Predictive Control* (MPC), disarankan untuk mengaplikasikan kontroler MPC ini dengan batasan atau *constraint*, baik constraint pada sinyal kontrol ataupun sinyal *output*. Pengaplikasi *constraint* pada kontroler MPC dapat mencegah *overshoot* dan sinyal kontrol berlebihan yang masuk ke dalam *driver* motor BLDC sehingga dapat mengurangi resiko rusaknya *driver* motor BLDC.

LAMPIRAN

A. Program Akuisisi Data Arduino.

```
#include <Servo.h>
Servo esc;
int setpoint;
int incomingByte;
int percenttrot;
int encoderPin = 7;
unsigned long duration;
unsigned long rpm;
char disp2[1];
char disp3[1];

void setup()
{
  Serial.begin(115200);
  pinMode(encoderPin,INPUT);
  esc.attach(9);
  esc.writeMicroseconds(1080);
  delay(2000);
}

void loop()
{
  setpoint = map(percenttrot,0,1000,1080,1750);
  esc.writeMicroseconds(setpoint);
  char disp1[10];
  if((setpoint<=1751)&&(setpoint>=1079))
  {
    sprintf(disp1,"*%4d",percenttrot);
    Serial.print(disp1);
  }
  else
  {
    setpoint=1080;
    percenttrot=0;
    sprintf(disp1,"*%4d",percenttrot);
```

```
    Serial.print(disp1);
  }
  if (Serial.available()>0)
  {
    percenttrot = Serial.parseInt();
  }
  duration = pulseIn(encoderPin,HIGH,100000);
  ram =(79550000/3)/duration;
  if (rpm>4000)
  {
    rpm = 0;
  }
  Serial.print('#');
  sprintf(disp2,"%4d",rpm);
  Serial.print(disp2);
  int sensorValue = analogRead(A0);
  int rem = map(sensorValue,0,1023,1,254);
  analogWrite(11,rem);
  int percentrem = map(sensorValue,0,1023,1,999);
  int arus = analogRead(A2)*4;
  sprintf(disp3,"%3d$%3d",percentrem,arus);
  Serial.print('@');
  Serial.print(disp3);
}
```

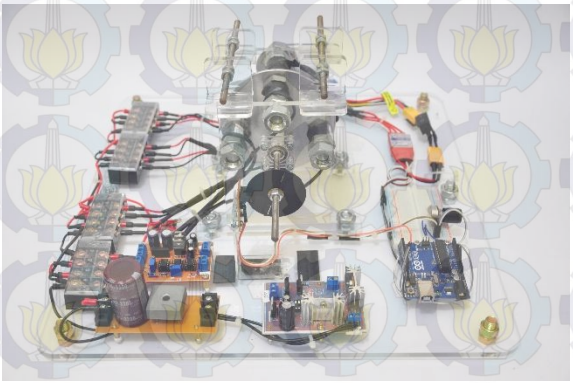
B. Program Pencarian Nilai *Root Mean Square Error* (RMSE).

```
clc;
n=size(output_40,1);
real=output_40(:,1);
model=output_40_model_arx3(:,1);
sum_err=0;
for i=1:n;
    if real(i)==0;
        real(i)=0.01;
        model(i)=0.01;
    end
    err=(real(i)-model(i))/real(i);
    sum_err=sum_err+err^2;
end
sum_err;
```

```
n;
rmse_val=sqrt(sum_err/n)
```

C. Hasil Perancangan Sistem Pengaturan Motor BLDC.

Perancangan sistem ini dikerjakan oleh BLDC Team yang beranggotakan 5 orang. Kelima orang tersebut adalah Fachrul Arifin, Beny Setyiadi Hidayat, Hudaibiy Hibban, Habib Ibnu Hasan dan Muhammad Ammar Huwaidi. Sistem dan *plant* ini kami beri nama BLDC V-1. Sistem ini dikerjakan selama satu semester dan dapat dijadikan acuan dan referensi sistem pada pengerjaan Tugas Akhir yang akan datang.



Gambar 1. *Plant* Sistem Pengaturan Kecepatan Motor BLDC dengan nama BLDC V-1

D. Program MATLAB Kontroler *Model Predictive Controller*.

```
clc;
clear;

%State space sistem
Ap=[0 1;0.005205 0.9936];
Bp=[0;1];
Cp=[0.006564 0.002823];
Dp=0;

%Parameter kontroler
```

```
Np=500; %prediction horizon
Nc=13; %control horizon
r_w=10; %tuning paramter

%Augmented model
[m1,n1]=size(Cp);
[n1,n_in]=size(Bp);
A_e=eye(n1+m1,n1+m1);
A_e(1:n1,1:n1)=Ap;
A_e(n1+1:n1+m1,1:n1)=Cp*Ap;
B_e=zeros(n1+m1,n_in);
B_e(1:n1,:)=Bp;
B_e(n1+1:n1+m1,:)=Cp*Bp;
C_e=zeros(m1,n1+m1);
C_e(:,n1+1:n1+m1)=eye(m1,m1);

%Matriks Phi dan F
n=n1+m1;
h(1,:)=C_e;
F(1,:)=C_e*A_e;
for kk=2:Np
    h(kk,:)=h(kk-1,:)*A_e;
    F(kk,:)= F(kk-1,:)*A_e;
end
v=h*B_e;
Phi=zeros(Np,Nc);
Phi(:,1)=v;
for i=2:Nc
    Phi(:,i)=[zeros(i-1,1);v(1:Np-i+1,1)];
end
BarRs=ones(Np,1);
Phi_Phi= Phi'*Phi; %Matriks Phi_Phi
Phi_F= Phi'*F; %Matriks Phi_F
Phi_R=Phi'*BarRs; %Matriks Phi_R
Q=zeros(1,Nc);
Q(1,1)=1;

%Mencari gain Kx,Ky
Kmpc=Q*inv(Phi_Phi+(r_w*eye(Nc,Nc)))*(Phi_F);
```

```
Ky=Q*inv(Phi_Phi+(r_w*eye(Nc,Nc)))*(Phi_R);
Kx=Kmpc(:,1:2);

%Mencari gain observer
pole=[-0.001 -0.12 -0.03];
Kob=place(A_e',C_e',pole)';
```

SPEED CONTROLLER DESIGN AND IMPLEMENTATION FOR BRUSHLESS DC MOTOR USING MODEL PREDICTIVE CONTROL (MPC)

Fachrul Arifin
2211100118

Advisor I : Ir. Josaphat Pramudijanto, M.Eng.
Advisor II : Ir. Ali Fatoni, M.T.

ABSTRACT

Nowadays, the research and development about electric vehicle becoming an interesting topic for researcher, students and global industry around the world. Most of the electric vehicle uses BLDC motor tchnology as their main powerdrive in the electric vehicle. As a future technology, electric vehicle have a high expectation as new transportation techonology to answer a global challenge and provide a better and greener transportation techonlogy around the world.

This Final Project will discuss about the BLDC Motor ability and speed response on load condition. On load condition, the speed response from BLDC motor will decrease and doesn't meet the criteria that we expected from. Because of that, the BLDC motor need some controller to eliminate the difference between the actual response and desired setpoint.

In this final project, the writer designed some controller based on Model Predictive Control (MPC) method to eliminate those error. The MPC method have some formula to predict the future behaviour of the plant output relies on the state-space model and current plant information. After some simulation and implementation, the MPC controller is capable to bringing back the actual response (in this case the speed of BLDC motor) to the desired setpoint with average 9.8% steady-state error in all load condition.

Keywords **:** *Electric Vehicle, Brushless DC, Model Predictive Control*

KATA PENGANTAR

Alhamdullilah, puji syukur penulis panjatkan kehadirat Allah SWT karena atas rahmat dan karunia-Nya penulis dapat menyelesaikan penulisan buku Tugas Akhir dengan judul **"PERANCANGAN DAN IMPLEMENTASI PENGATURAN KECEPATAN MOTOR *BRUSHLESS DC* MENGGUNAKAN METODE *MODEL PREDICTIVE CONTROL* (MPC)"**. Tugas akhir merupakan salah satu syarat yang harus dipenuhi untuk menyelesaikan program studi Strata-1 pada Jurusan Teknik Elektro Fakultas Teknologi Industri Institut Teknologi Sepuluh Nopember Surabaya.

Penulis menyadari bahwa dalam penulisan skripsi ini banyak mengalami kendala, namun berkat bantuan, bimbingan, dan kerjasama dari berbagai pihak sehingga kendala-kendala tersebut dapat diatasi. Untuk itu pada kesempatan ini penulis menyampaikan banyak terimakasih dan penghargaan setingi-tingginya kepada :

1. Kedua orang tua, Ayahanda Piping Zaenal Aripin dan Ibunda Sofiarini serta adikku tercinta Lina Maghfira yang selalu memberikan dukungan, semangat, dan doa kepada penulis.
2. Bapak Josaphat Pramudijanto dan Bapak Ali Fatoni selaku Dosen Pembimbing atas segala bantuan, perhatian, dan arahan selama pengerjaan Tugas Akhir ini.
3. Bapak Rusdhianto Effendie selaku Koordinator Bidang Studi Teknik Sistem Pengaturan Jurusan Teknik Elektro ITS dan Bapak Tri Arief Sardjono selaku Ketua Jurusan Teknik Elektro ITS.
4. Bapak dan Ibu dosen bidang studi Teknik Sistem Pengaturan, Teknik Elektro ITS.
5. Beny, Habib, Huda dan Ammar sebagai BLDC Team yang sangat solid dalam mendesain dan mewujudkan *plant* BLDC.
6. Mas Fahrul Abbas sebagai pembimbing S2 yang sangat membantu dalam mewujudkan berdirinya *plant* BLDC ini.
7. Serta rekan-rekan yang tidak saya bisa sebutkan satu-persatu.

Penulis menyadari bahwa pada penyusunan laporan tugas akhir ini masih terdapat kekurangan-kekurangan karena keterbatasan kemampuan yang penulis miliki, walaupun demikian penulis berharap tugas akhir ini dapat bermanfaat bagi yang membutuhkannya.

Surabaya, Juli 2015

Penulis

DAFTAR ISI

HALAMAN

HALAMAN JUDUL ..... **i**
PERNYATAAN KEASLIAN TUGAS AKHIR ..... **v**
LEMBAR PENGESAHAN ..... **vii**
ABSTRAK ..... **ix**
ABSTRACT ..... **xi**
KATA PENGANTAR ..... **xiii**
DAFTAR ISI ..... **xv**
DAFTAR GAMBAR ..... **xix**
DAFTAR TABEL ..... **xxi**

BAB 1. PENDAHULUAN **1**
1.1 Latar Belakang 1
1.2 Perumusan Masalah 2
1.3 Batasan Masalah 2
1.4 Tujuan Penelitian 2
1.5 Sistematika Penulisan 3
1.6 Relevansi 4

BAB 2. TEORI PENUNJANG **5**
2.1 Motor *Brushless* DC 5
2.1.1 Konstruksi Motor *Brushless* DC 5
2.1.2 *Driver* dan Kontroler Motor *Brushless* DC 8
2.1.3 Prinsip Kerja Motor *Brushless* DC 9
2.2 Rem Elektromagnetik 10
2.3 Sensor *Rotary Encoder* 11
2.4 Arduino Uno 12
2.5 MATLAB R2014a 13
2.6 Identifikasi Sistem 14
2.6.1 Identifikasi Dinamis 15
2.6.1.1 Sinyal *Pseudo-Random Binary Sequence* 16
2.6.1.2 *Auto-Regressive Exogeneous* (ARX) 16
2.6.2 Validasi Model 17

2.7 *Model Predictive Control* (MPC) 18
2.7.1 Model *State-Space* dengan *Embedded Integrator* 19
2.7.2 Perhitungan *Prediction Output* & *Future Control* 20
2.7.3 Indeks Performansi Kontroler MPC 21
2.7.4 *Closed-loop Control System* 22
2.7.5 *State Estimation* 24

BAB 3. PERANCANGAN SISTEM 27
3.1 Gambaran Umum Sistem 27
3.2 Perancangan Perangkat Keras 28
3.2.1 Perancangan Mekanik 28
3.2.1.1 Motor *Brushless DC* 28
3.2.1.2 *Electronic Speed Control* (ESC) 29
3.2.1.3 Rem Elektromagnetik 31
3.2.2 Perancangan Elektronik 32
3.2.2.1 Rangkaian Sensor *Rotary Encoder* 32
3.2.2.2 Rangkaian Penyearah Gelombang AC 33
3.2.2.3 Rangkaian *Driver* Rem Elektromagnetik 34
3.2.2.4 Rangkaian Sensor Arus 35
3.2.2.5 Mikrokontroler Arduino 36
3.3 Perancangan Perangkat Lunak 37
3.3.1 *Software* Arduino 37
3.3.2 *Software* MATLAB 37
3.4 Identifikasi dan Pemodelan Sistem 39
3.4.1 Metode Pembebanan *Plant* 39
3.4.2 Metode Identifikasi dan Pemodelan 39
3.4.3 Pemodelan Motor *Brushless* DC 40
3.4.3.1 Beban Minimal 40
3.4.3.2 Beban Nominal 41
3.4.3.3 Beban Maksimal 42
3.4.4 Pengujian dan Validasi 42
3.5 Perancangan Kontroler *Model Predictive Control* 43
3.5.1 Perancangan Model *State Space* 43
3.5.2 Desain *Augmented Model* 44
3.5.3 Penentuan Parameter dan *Gain* Kontroler MPC 45
3.5.4 Desain *State Estimation* Menggunakan *Observer* 47

BAB 4. PENGUJIAN DAN ANALISA 49
4.1 Gambaran Umum Pengujian Sistem 49
4.2 Pengujian Perangkat Keras 50
4.2.1 Pengujian *Electronic Speed Control* Motor BLDC 50
4.3 Simulasi Sistem 51
4.3.1 Diagram Blok Simulasi Sistem 51
4.3.2 Prosedur Pengerjaan Simulasi Sistem 53
4.4 Pengaruh Parameter N_p pada Sistem 53
4.4.1 Beban Minimal 55
4.4.2 Beban Nominal 55
4.4.3 Beban Maksimal 56
4.5 Pengaruh Parameter N_c pada Sistem 57
4.5.1 Beban Minimal 57
4.5.2 Beban Nominal 59
4.5.3 Beban Maksimal 60
4.6 Pengaruh Parameter r_w pada Sistem 60
4.6.1 Beban Minimal 62
4.6.2 Beban Nominal 62
4.6.3 Beban Maksimal 63
4.7 Respon *Tracking* Sistem 64
4.7.1 Beban Minimal 64
4.7.2 Beban Nominal 65
4.7.3 Beban Maksimal 66
4.8 Implementasi Sistem 67
4.8.1 Realisasi Perancangan Sistem 67
4.8.2 Diagram Blok Implementasi Sistem 71
4.8.3 Analisa Kontroler MPC pada Implementasi Sistem 73
4.8.3.1 Beban Minimal 74
4.8.3.2 Beban Nominal 75
4.8.3.3 Beban Maksimal 77

BAB 5. PENUTUP 79
5.1 Kesimpulan 79
5.2 Saran 80

DAFTAR PUSTAKA 81
LAMPIRAN 83
RIWAYAT HIDUP 89

DAFTAR TABEL

Tabel 3.1 Spesifikasi Motor BLDC RCTimer HP2212-1000KV..... 29
Tabel 3.2 Spesifikasi ESC Turnigy PLUSH-25A. 30
Tabel 3.3 Spesifikasi Sensor Arus SparkFun ACS712 5A. 36
Tabel 3.4 Metode Pembebanan pada Sistem. 39
Tabel 3.5 Identifikasi Dinamis pada Beban Minimal. 41
Tabel 3.6 Identifikasi Dinamis pada Beban Nominal. 42
Tabel 3.7 Identifikasi Dinamis pada Beban Maksimal. 42
Tabel 3.8 Pemodelan Motor BLDC pada Berbagai Macam Kondisi Pembebanan 43
Tabel 4.1 Hubungan Nilai *Input* PWM, Tegangan *Output* ESC dan Kecepatan Motor BLDC. 51
Tabel 4.2 Karakteristik Respon Sistem pada Pembebanan Minimal untuk Tiap-Tiap Variabel *Prediction Horizon*. 55
Tabel 4.3 Karakteristik Respon Sistem pada Pembebanan Nominal untuk Tiap-Tiap Variabel *Prediction Horizon*. 56
Tabel 4.4 Karakteristik Respon Sistem pada Pembebanan Maksimal untuk Tiap-Tiap Variabel *Prediction Horizon*. 56
Tabel 4.5 Karakteristik Respon Sistem pada Pembebanan Minimal untuk Tiap-Tiap Variabel *Control Horizon*..... 59
Tabel 4.6 Karakteristik Respon Sistem pada Pembebanan Nominal untuk Tiap-Tiap Variabel *Control Horizon*..... 59
Tabel 4.7 Karakteristik Respon Sistem pada Pembebanan Maksimal untuk Tiap-Tiap Variabel *Control Horizon*..... 60
Tabel 4.8 Karakteristik Respon Sistem pada Pembebanan Minimal untuk Tiap-Tiap Variabel *Tuning Parameter*..... 62
Tabel 4.9 Karakteristik Respon Sistem pada Pembebanan Nominal untuk Tiap-Tiap Variabel *Tuning Parameter*..... 63
Tabel 4.10 Karakteristik Respon Sistem pada Pembebanan Maksimal untuk Tiap-Tiap Variabel *Tuning Parameter*..... 63
Tabel 4.11 Karakteristik Respon Sistem pada Pembebanan Minimal dengan Referensi *Tracking*..... 64
Tabel 4.12 Karakteristik Respon Sistem pada Pembebanan Nominal dengan Referensi *Tracking*..... 65
Tabel 4.13 Karakteristik Respon Sistem pada Pembebanan Maksimal dengan Referensi *Tracking*..... 66

Tabel 4.14 Nilai Parameter dari Kontroler MPC untuk Tiap–Tiap Nilai Pembebanan. 73
Tabel 4.15 Karakteristik Respon dengan Pembebanan Minimal pada Implementasi Sistem. 74
Tabel 4.16 Karakteristik Respon dengan Pembebanan Nominal pada Implementasi Sistem. 76
Tabel 4.17 Karakteristik Respon dengan Pembebanan Maksimal pada Implementasi Sistem 78

DAFTAR GAMBAR

Gambar 2.1 Konstruksi Umum Motor *Brushless* DC (BLDC)..... 6
Gambar 2.2 Konstruksi *Brushless* DC (BLDC) Tipe *Outer Rotor*.... 6
Gambar 2.3 *Timing* Komutasi *Stator* pada Motor BLDC..... 7
Gambar 2.4 Skema Rangkaian *Driver* & Kontroler Motor BLDC.... 8
Gambar 2.5 Prinsip Kerja Motor BLDC..... 10
Gambar 2.6 Struktur dari Sebuah Rem Elektromagnetik. 11
Gambar 2.7 Konstruksi dan Prinsip Kerja Sensor *Rotary Encoder*. 12
Gambar 2.8 Dialog Tampilan *Instrument Control Toolbox*..... 14
Gambar 2.9 Sinyal Uji *Pseudo-Random Binary Sequence* (PRBS). 17
Gambar 2.10 Konsep dari Kontroler *Model Predictive Control*..... 18
Gambar 2.11 Diagram Blok Sistem Kontrol *Loop* Tertutup *Model Predictive Control* untuk Waktu Diskrit..... 23
Gambar 2.12 Diagram Blok Sistem Kontrol *Loop* Tertutup *Model Predictive Control* untuk Waktu Diskrit dengan menggunakan *Observer*..... 25
Gambar 3.1 Blok Diagram Sistem Pengaturan Kecepatan Motor *Brushless DC* (BLDC)..... 27
Gambar 3.2 Motor *Brushless* DC Tipe *Outrunner* dengan Tipe RCTimer HP2212-1000KV..... 29
Gambar 3.3 *Electronic Speed Control* Turnigy PLUSH-25A. 30
Gambar 3.4 Perancangan Konstruksi Fisik Rem Elektromagnetik.. 31
Gambar 3.5 Sensor *Rotary Encoder* yang Terpasang pada *Shaft*. ... 32
Gambar 3.6 Skema Rangkaian *Rotary Encoder* yang Tersusun dari *Optocoupler* dan Resistor. 33
Gambar 3.7 Rangkaian Penyearah Gelombang AC yang Terdiri dari *Diode Bridge* dan Kapasitor. 33
Gambar 3.8 Skema Rangkaian *Driver* Rem Elektromagnetik..... 34
Gambar 3.9 Sensor Arus SparkFun ACS712 5A Breakout 35
Gambar 3.10 Mikrokontroler Arduino Uno R3..... 36
Gambar 3.11 Tampilan Arduino IDE 38
Gambar 3.12 Blok Identifikasi Sistem *Open Loop* pada Simulink.... 38
Gambar 3.13 Hubungan *Input-Output* Sistem pada Identifikasi Dinamis..... 40
Gambar 3.14 Perbandingan Respon Hasil Pengukuran dan Pemodelan pada Beban Minimal..... 41

Gambar 4.1 Realisasi *Plant* atau Sistem Pengaturan Kecepatan Motor *Brushless DC*. 49
Gambar 4.2 Diagram Blok Simulasi Pengujian Kontroler MPC pada Sistem..... 52
Gambar 4.3 Subsistem *Observer* pada Simulasi Sistem 52
Gambar 4.4 Pengaruh Parameter *Prediction Horizon* pada Sistem.. 54
Gambar 4.5 Sinyal Kontrol pada Sistem dengan Perubahan Parameter *Prediction Horizon*. 54
Gambar 4.6 Pengaruh *Control Horizon* pada Sistem. 58
Gambar 4.7 Sinyal Kontrol pada Sistem dengan Perubahan Parameter *Control Horizon*. 58
Gambar 4.8 Pengaruh Parameter *Tuning* Indeks Performansi pada Sistem..... 61
Gambar 4.9 Sinyal Kontrol pada Sistem dengan Perubahan *Tuning* Parameter pada Indeks Performansi. 61
Gambar 4.10 Respon Sistem pada Pembebanan Minimal dengan Referensi *Tracking*. 65
Gambar 4.11 Respon Sistem pada Pembebanan Nominal dengan Referensi *Tracking*. 66
Gambar 4.12 Respon Sistem pada Pembebanan Maksimal dengan Referensi *Tracking*. 67
Gambar 4.13 Realisasi Perancangan Sistem Pengaturan Kecepatan Motor BLDC. 68
Gambar 4.14 Konstruksi Rem Elektromagnetik pada Sistem. 69
Gambar 4.15 Hasil Perancangan Sensor *Rotary Encoder*. 69
Gambar 4.16 Alat Akuisisi Data berupa Mikrokontroler Arduino..... 70
Gambar 4.17 Rangkaian *Rectifier* pada Sistem..... 71
Gambar 4.18 Rangkaian *Driver* untuk Pengaturan Tegangan *Input* pada Rem Elektromagnetik. 71
Gambar 4.19 Diagram Blok Implementasi Kontroler MPC pada Sistem..... 72
Gambar 4.20 Subsistem *Plant* yang Terdiri Dari Komunikasi *Serial* pada Sistem. 72
Gambar 4.21 Hasil Implementasi Kontroler MPC pada Sistem dengan Pembebanan Minimal. 75
Gambar 4.22 Hasil Implementasi Kontroler MPC pada Sistem dengan Pembebanan Nominal..... 76
Gambar 4.23 Hasil Implementasi Kontroler MPC pada Sistem dengan Pembebanan Maksimal..... 77

RIWAYAT HIDUP

Fachrul Arifin, lahir di Cimahi, 10 April 1993. Riwayat pendidikannya, menamatkan pendidikan dasar di SD Negeri 1 Rancabelut Cimahi (tahun 2005), pendidikan menengah di SMP Negeri 1 Cimahi (tahun 2008), dan pendidikan tinggi di SMA Negeri 2 Bandung (tahun 2011). Saat ini telah menempuh kuliah di Jurusan Teknik Elektro Institut Teknologi Sepuluh Nopember dengan mengabil bidang studi Teknik Sistem Pengaturan. Selama kuliah penulis aktif dalam beberapa kegiatan akademis maupun non akademis. Penulis aktif dalam Himpunan Mahasiswa Teknik Elektro bidang Lingkar Kampus dan pernah menjabat sebagai Sekreteris Departemen periode 2013-2014. Penulis juga aktif dalam kegiatan keilmiahan seperti seminar dan berbagai macam pelatihan bertema keilmiahan. Penulis dapat dihubungi melalui email: **fachrul.arifin@live.com**

BAB 1
PENDAHULUAN

Bab 1 pada laporan Tugas Akhir ini akan menjelaskan latar belakang yang akan diangkat menjadi suatu rumusan masalah pada Tugas Akhir kali ini. Setelah rumusan masalah terbentuk, Bab 1 juga akan menjelaskan mengenai tujuan dan batasan masalah yang akan dikerjakan pada Tugas Akhir kali ini. Terakhir, pada Bab 1 juga dijelaskan mengenai sistematika penulisan dan relevansi pengerjaan Tugas Akhir ini.

1.1 Latar Belakang

Saat ini, seperti kita telah ketahui bersama bahwa cadangan dan persediaan bahan bakar fosil dari tahun ke tahun cenderung menipis. Kabar ini tentu saja menjadi hal yang buruk bagi dunia otomotif yang mengandalkan konsumsi bahan bakar fosil yang saat ini menjadi sumber utama penggerak kendaraan bermotor. Tentu saja, hal ini mendorong umat manusia untuk mencari bahan bakar alternatif dan lebih efisien dibandingkan bahan bakar fosil. Saat ini, telah banyak ditemukan berbagai macam kendaraan bermotor berbasiskan energi alternatif yang lebih murah dan efisien dibandingkan bahan bakar fosil. contohnya adalah mobil listrik yang saat ini menjadi tren dan diproyeksikan menjadi kendaraan masa depan.

Kendaraan listrik merupakan kendaraan yang menggunakan murni energi listrik sebagai penggeraknya. Pada kendaraan konvensional, bahan bakar dari minyak bumi yang telah diproses, contohnya premium dan pertama, digunakan untuk menjalankan mesin diesel biasa. Sedangkan pada kendaraan listrik, digunakan sumber energi listrik yang berasaldari baterai untuk menggerakkan kendaraan listrik tersebut.

Salah satu penggerak utama dari kendaraan listrik adalah Motor *Brushless* DC (BLDC) yang berfungsi sebagai komponen utama penggerak mobil listrik ini. Motor BLDC merupakan pengembangan dari motor DC konvensional atau *brushed* DC motor. Pada mobil konvensional berbahan bakar fosil, motor BLDC banyak digunakan dalam komponen penyusun mobil, seperti *Air Conditioner* (AC), *airbag*, *wiper blades*, dan macam lainnya. Kedepannya, motor BLDC diharapkan dapat menjadi penggerak utama pada mobil listrik walaupun kondisi jalan yang menantang sekalipun [1].

Pada rencana tugas akhir kali ini, akan diterapkan sebuah kontroler berbasiskan *Model Predictive Control* (MPC) untuk dapat menggerakkan motor BLDC secara optimal pada kondisi berbeban. MPC dipilih karena konsepnya yang sangat intuitif dan penalaannya yang mudah. Selain itu, MPC juga dapat menangani sistem dengan memperhitungkan batasan atau *constraint* [1]. Harapan dari implementasi kontroler MPC pada motor BLDC adalah didapatkannya respon motor yang cepat dan sesuai dengan kebutuhan mobil listrik.

1.2 Perumusan Masalah

Apabila suatu motor BLDC tidak diberikan beban, tentu saja kecepatan motor akan berjalan konstan dan *transfer function* dari motor BLDC akan berjalan dengan semestinya. Akan tetapi, di saat motor BLDC diberikan suatu beban, maka kecepatan motor akan berubah dan membuat *transfer function* motor bernilai tidak linear. Efek pembebanan ini dapat terjadi dikarenakan kondisi beban yang berubah-ubah pada mobil, contohnya akibat penambahan beban pada konstruksi mobil ataupun beban penumpang yang semakin bertambah. Tentu saja efek ini tidak diharapkan dikarenakan dapat mengganggu kinerja dan performa berupa penurunan kecepatan dari motor BLDC. Kontroler *Model Predictive Control* ini diharapkan dapat mengurangi dan mengeliminasi *error* yang timbul akibat perubahan beban yang ditanggung Motor BLDC.

1.3 Batasan Masalah

Berdasarkan rumusan masalah, penulis akan membatasi permasalahan yang akan diteliti sehingga tujuan dari penelitian dapat dicapai. Batasan dari penelitian tersebut adalah sebagai berikut:

a. *Range* kecepatan dari motor BLDC yang akan diteliti berada pada *range* kecepatan 1500-2500 RPM.
b. Metode yang akan digunakan pada sistem pengaturan kecepatan motor BLDC adalah *Model Predictive Control* menggunakan mikrokontroler Arduino Uno dan *software* MATLAB.
c. Penambahan beban yang akan digunakan pada motor BLDC menggunakan rem elektromagnetik dengan 3 variabel beban, yaitu minimal, nominal dan maksimal.

1.4 Tujuan Penelitian

Tujuan dari pelaksanaan Tugas Akhir ini adalah merancang dan mengimplementasikan suatu sistem pengaturan kecepatan motor BLDC

dengan menggunakan metode *Model Predictive Control* untuk menghilangkan *error* penurunan kecepatan akibat adanya beban.

1.5 Sistematika Penulisan

Sistematika penulisan pada laporan Tugas Akhir ini terdiri atas 5 bab, seperti yang dapat di lihat pada uraian berikut ini :

BAB 1 : PENDAHULUAN

Pada bab ini, akan dijelaskan mengenai latar belakang serta perumusan dan batasan masalah pada Tugas Akhir ini. Selain itu, akan dijabarkan pula tujuan dari Tugas Akhir ini beserta metodologi yang digunakan. Terakhir, akan dijelaskan pula mengenai sistematika penulisan dan relevansi Tugas Akhir ini.

BAB 2 : TEORI PENUNJANG

Bab ini menjelaskan tentang landasan teori yang diadopsi dan dipelajari pada pelaksanaan Tugas Akhir ini. Materi yang akan dijelaskan antara lain teori mengenai motor BLDC beserta perangkat keras pendukungnya serta metode dan perangkat lunak yang digunakan untuk pengaturan kecepatan motor BLDC.

BAB 3 : PERANCANGAN SISTEM

Bab ini menjelaskan tentang perancangan keras dan perangkat lunak yang akan digunakan pada pelaksanaan Tugas Akhir. Selain itu, akan dijabarkan pula mengenai perancangan kontroler menggunakan metode *Model Predictive Control* (MPC).

BAB 4 : PENGUJIAN DAN ANALISA

Pada bab ini, akan dijabarkan mengenai hasil simulasi kontroler pada *software* beserta analisanya. Selain itu, bab ini juga berisi tentang hasil implementasi kontroler pada *plant* motor BLDC beserta analisanya.

BAB 5 : PENUTUP

Bab terakhir ini akan menjelaskan tentang penarikan kesimpulan pelaksanaan Tugas Akhir dan saran untuk penelitian selanjutnya.

1.6 Relevansi

Hasil yang diperoleh dari pelaksanaan Tugas Akhir ini diharapkan menjadi referensi perancangan kontroler untuk mengatur kecepatan dari motor BLDC pada skala kendaraan listrik sesungguhnya. Selain itu, hasil dari penelitian ini juga diharapkan menjadi referensi perancangan kontroler MPC untuk pengaturan kecepatan jenis motor lainnya.

BAB 2
TEORI PENUNJANG

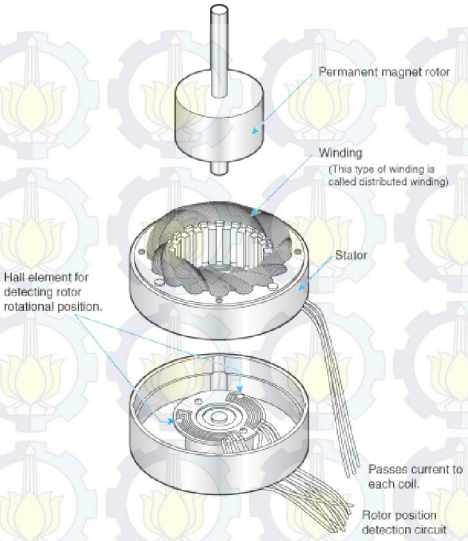
Bab 2 pada laporan Tugas Akhir ini akan menjelaskan mengenai topik dan teori umum yang menyangkut pelaksanaan Tugas Akhir kali ini. Pada bagian awal akan dijelaskan mengenai konstruksi dan cara kerja motor BLDC. Setelah itu, dideskripsikan pula mengenai identifikasi dan pemodelan suatu sistem menggunakan identifikasi dinamis. Terakhir, akan dijelaskan mengenai struktur dari kontroler yang berbasis *Model Predictive Control* atau MPC.

2.1 Motor *Brushless* DC

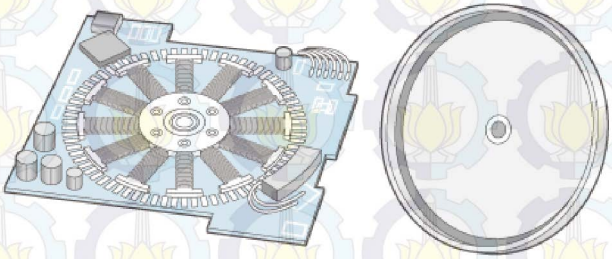
Motor *Brushless* DC (BLDC) merupakan salah satu jenis motor DC yang konstruksinya menggunakan magnet permanen di bagian *rotor* dan Kumparan jangkar pada *stator*. Motor BLDC tidak menggunakan fungsi *brush* sebagai media eksitasi ke *rotor*-nya, namun fungsi tersebut digantikan oleh medan magnet yang telah ditimbulkan oleh magnet permanen pada bagian *rotor*-nya. Keuntungan yang paling jelas dari konfigurasi *brushless* adalah penghapusan *brush*, yang memudahkan perawatan dan menghilangkan rugi gesek akibat adanya kontak antara *rotor* dan *brush*. Motor BLDC lebih baik dibandingkan dengan motor induksi, motor BLDC memiliki efisiensi yang lebih baik dan faktor daya yang lebih baik dan, oleh karena itu, daya *output* yang lebih besar untuk *frame* yang sama. [2]

2.1.1 Konstruksi Motor *Brushless* DC

Seperti yang telah dijelaskan pada paragraf sebelumnya, perbedaan mendasar antara motor DC konvensional dan motor BLDC adalah pada media eksitasinya. Untuk menggantikan peran *brush* seperti yang terdapat pada motor DC konvensional, digunakanlah rangkaian penggerak (*driver*) yang terdiri dari komponen semikonduktor atau MOSFET untuk menjalankan pensaklaran arus pada kumparan motor BLDC. Tapi rangkaian *driver* tersebut, motor BLDC tidak akan dapat berjalan sebagaimana mestinya. Secara garis besar, konstruksi umum motor BLDC dapat dilihat pada Gambar 2.1 dan Gambar 2.2. Terlihat pada gambar, motor BLDC terdiri dari 3 bagian utama, yaitu *rotor* yang berupa magnet permanen, kumparan jangkar atau *stator*, dan sensor posisi *rotor* yang berupa berupa sensor *hall-effect*.



Gambar 2.1 Konstruksi Umum Motor *Brushless* DC (BLDC). [3]



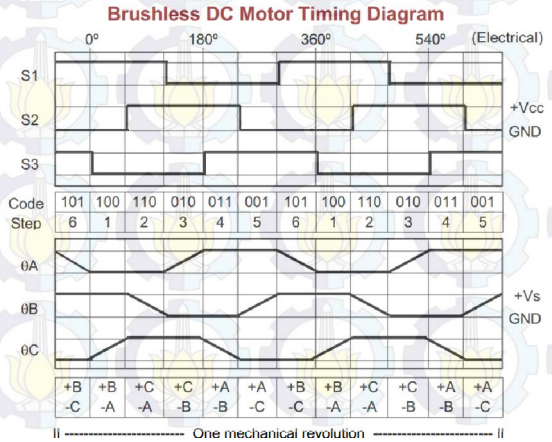
Gambar 2.2 Konstruksi *Brushless* DC (BLDC) Tipe *Outer Rotor*. [3]

Pada Tugas Akhir kali ini, akan dijelaskan mengenai konstruksi motor BLDC yang digunakan, yaitu motor BLDC tipe *radial flux* dan *outer rotor*. Motor yang digunakan pada Tugas Akhir ini dirancang untuk mengalirkan fluks secara *radial*. Dan konstruksi motor BLDC yang digunakan adalah tipe *outer rotor*. Tipe *outer rotor* memiliki kumparan

jangkar yang diletakkan pada bagian dalam motor. Sedangkan bagian *rotor* diletakkan pada bagian luar motor. Konstruksi motor BLDC tipe *outer rotor* dapat dilihat pada Gambar 2.2.

Stator pada motor BLDC merupakan rangkaian yang terdiri dari beberapa belitan yang disebut kumparan jangkar. Jumlah pasang kutub yang diciptakan di *stator* akibat dari arah dan jumlah lilitan berpengaruh terhadap performa dari motor BLDC. *Stator* pada motor BLDC terdiri dari 3 fasa yang biasanya terhubung dengan tiga buah kabel untuk disambungkan pada rangkaian kontrolnya. *Rotor* pada motor BLDC terdiri dari beberapa magnet permanen. Jumlah kutub magnet di *rotor* juga mempengaruhi ukuran langkah dan torsi dari motor. Magnet permanen pada motor BLDC biasanya menggunakan jenis magnet neodymium yang merupakan jenis magnet tetap yang sangat kuat.

Untuk menentukan orientasi posisi *rotor* fungsi komutasi dilakukan oleh sensor, terdapat beberapa jenis sensor yang digunakan. Misalkan sensor *optical encoder, magnetic encoder* atau *hall effect* yang berfungsi memberikan sinyal digital akibat adanya medan magnetik yang tegak lurus terhadap sensor. Sensor *hall* ini harus diletakkan sedekat mungkin dengan *rotor* magnet permanen untuk mendeteksi posisi kutub magnet pada *rotor*. Output *hall* akan akan dibaca oleh *decoder* yang nanti akan menentukan *timing* komutasi seperti pada Gambar 2.3 dan menentukan kumparan *stator* mana yang akan dinyalakan sehingga motor BLDC dapat berputar.

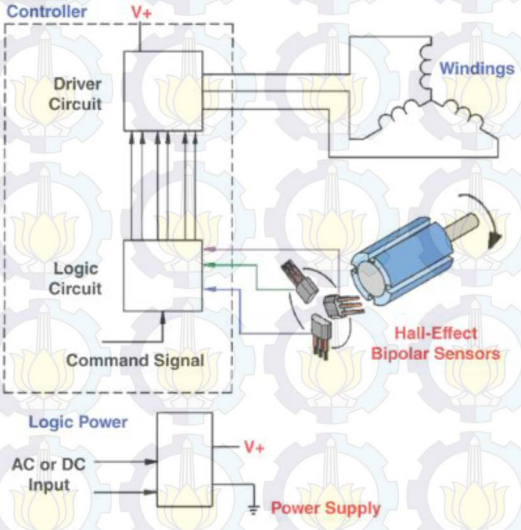


Gambar 2.3 *Timing* Komutasi *Stator* pada Motor BLDC. [4]

2.1.2 *Driver* dan Kontroler Motor *Brushless* DC

Driver atau kontroler pada motor BLDC sangatlah penting karena rangkaian inilah yang menggantikan peran *brush* pada motor DC konvensional untuk menyegerakan stator. Motor BLDC memerlukan suatu *trigger* pulsa yang masuk ke bagian stator motor BLDC untuk memberikan pengaturan besarnya arus yang mengalir sehingga motor dapat diatur secara akurat. Selain itu, *driver* pada motor BLDC juga berperan untuk mengubah tegangan DC menjadi tegangan AC dikarenakan motor BLDC memiliki kumparan stator 3 fasa pada konstruksinya. Secara umum, skema rangkaian kontrol dan *driver* pada motor BLDC dapat dilihat pada Gambar 2.4.

Pada Gambar 2.4, terlihat bahwa kontroler motor BLDC terdiri dari 2 bagian, yaitu rangkaian *driver* (*driver circuit*) dan rangkaian logika (*logic circuit*). Rangkaian *driver* biasanya terdiri dari beberapa transistor MOSFET (*metal–oxide–semiconductor field-effect transistor*) atau IGBT (*insulated-gate bipolar transistor*) yang berfungsi untuk merubah arus DC yang diberikan rangkaian logika dan mengubahnya menjadi arus AC dengan perbedaan fasa 120^{o} untuk menjalankan motor BLDC.



Gambar 2.4 Skema Rangkaian *Driver* & Kontroler Motor BLDC. [5]

Pada kontroler motor BLDC, rangkaian logika mempunyai peranan yang sangat penting untuk mengatur *timing* komutasi dan waktu *switching* yang tepat pada tiap fasa *stator* motor BLDC. Rangkaian logika ini bisa berupa mikroprosesor atau mikrokontroler yang menerima *input* dari sensor *hall effect* pada motor atau tegangan *back-emf* pada motor. *Input* ini akan diolah sedemikian rupa sehingga rangkaian logika ini dapat memberi *output* perintah kepada rangkaian *driver* untuk menyalakan kumparan stator tertentu pada motor BLDC. Saat ini, banyak digunakan jenis kontroler modern yang menggunakan mikrokontroler dengan logika tertentu, dengan *decoder* akan mengatur *switching* transistor sehingga terbentuk pola *switching* yang tepat pada tiap fase untuk mengelola akselerasi, kontrol kecepatan dan menyempurnakan efisiensi.

2.1.3 Prinsip Kerja Motor *Brushless* DC

Prinsip kerja mendasar dari motor BLDC adalah teori mengenai medan magnet, saat suatu kutub utara dengan medan magnet yang keluar akan saling tolak menolak dengan kutub yang sejenisnya begitupun sebaliknya akan saling tarik menarik jika magnetnya berlawan kutub. Dari prinsip sederhana diatas dapat diterapkan dalam penggunaan sistem kerja motor BLDC, yang memiliki medan magnet permanen pada *rotor* dan gaya elektromagnet (magnet yang ditimbulkan dari pemberian input arus listrik) pada bagian kumparan *stator*. Pada motor BLDC, fungsi pengaturan terhadap inputan arus yang harus diberikan ke kumparan stator untuk dapat menimbulkan medan elektromagnet yang tepat guna memutar rotor dilakukan oleh kontroler. Elemen inilah yang menjadi elemen utama yang membedakannya dengan motor DC konvensional, dan menggantikan kerja komutasi mekanisnya.

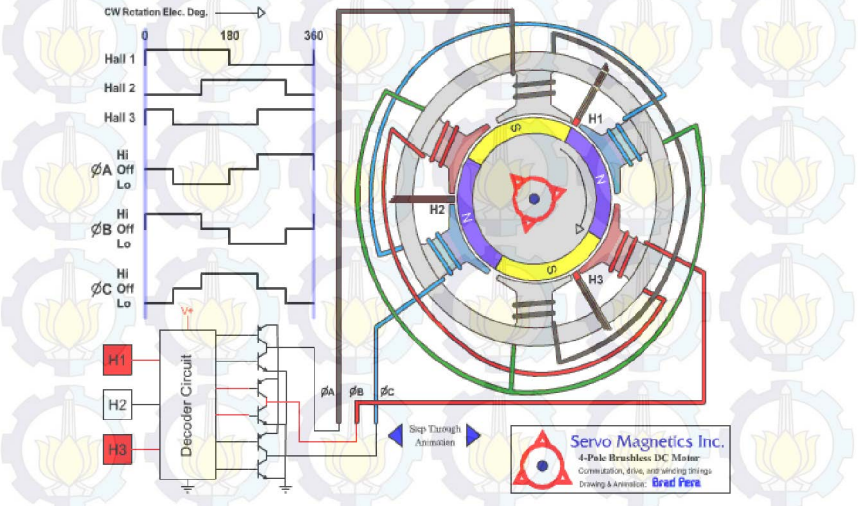
Gambar 2.5 memperlihatkan prinsip kerja motor BLDC 3 fasa dan 2 pasang kutub *rotor* secara umum. Terlihat pada gambar bahwa pendeteksian posisi *rotor* menggunakan 3 buah sensor *hall effect* H1, H2, dan H3 yang diletakkan pada ujung plat dan membentuk lingkaran dengan interval 120°. Pada gambar yang pertama, H2 mendeteksi fluks magnetik kutub utara dan *decoder* mengaktifkan B+ C-. Dalam kondisi ini, arus mengalir ke kumparan stator B dan membentuk kutub utara. Arus juga akan mengalir ke kumparan stator C dan membentuk kutub selatan. Akibatnya terjadi gaya tolak menolak antara *stator* dan *rotor* sehingga *rotor* akan berputar searah jarum jam. Begitu seterusnya kutub *rotor* akan berjalan sesuai dengan pensaklaran yang terus beganti. Dengan mengulang proses pensaklaran sesuai urutan seperti terlihat pada Gambar

2.5, maka *rotor* yang terdiri dari magnet permanen akan berputar secara terus menerus.

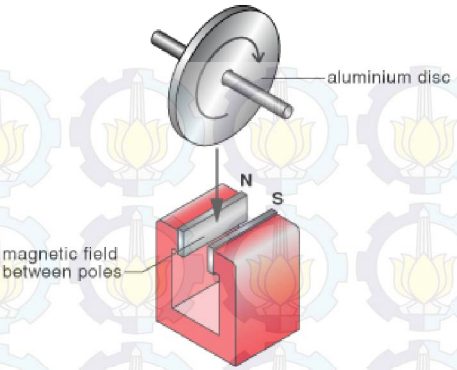
2.2 Rem Elektromagnetik

Sistem pengereman ini menggunakan gaya elektromagnetik yang timbul dari suatu magnet permanen tetap atau kumparan yang diberikan arus listrik untuk memperlambat suatu gerakan. Konstruksi dasarnya berupa suatu piringan logam non-ferromagnetik yang terpasang pada suatu poros yang berputar. Piringan logam tersebut diapit oleh kumparan yang dialiri arus listrik hingga menimbulkan medan magnet yang kutubnya saling berlawanan. Logam piringan tersebut akan memotong medan magnet yang ditimbulkan oleh kumparan tersebut sehingga menimbulkan *eddy current* atau arus *eddy*.

Arus *eddy* merupakan arus listrik yang timbul bilamana suatu piringan logam berada di sekitar medan magnet yang garis-garis gayanya sedang berubah-ubah. Arus *eddy* ini mempunyai medan magnet yang arahnya berlawanan dengan arah gerak piringan logam. Akibatnya laju piringan logam akan tertahan akibat dari adanya arus *eddy* ini. Gambar 2.6 akan memperlihatkan struktur dan konstruksi dari sebuah sistem rem elektromagnetik.



Gambar 2.5 Prinsip Kerja Motor BLDC. [6]



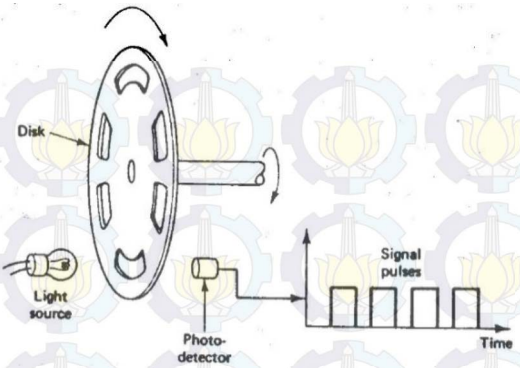
Gambar 2.6 Struktur dari Sebuah Rem Elektromagnetik. [7]

Rem elektromagnetik biasa diletakkan dekat dengan bagian yang bergerak. Rem ini bekerja pada kondisi yang dingin dan memenuhi persyaratan energi pengereman kecepatan tinggi karena tanpa adanya gesekan. Selain pada kendaraan listrik, aplikasi rem elektromagnetik juga dapat ditemukan pada sistem pengereman kereta api.

2.3 Sensor *Rotary Encoder*

Sensor *rotary encoder* adalah sebuah sensor elektromekanik yang dapat memonitor suatu gerakan atau keadaan posisi. Sensor ini umumnya menggunakan sensor optik untuk menghasilkan pulsa yang dapat diterjemahkan menjadi gerakan, posisi dan arah. Posisi sudut poros benda yang berputar dapat diolah menjadi informasi berupa kode digital oleh sensor *rotary encoder* untuk digunakan sebagai *feedback* dan diteruskan ke rangkaian kendali. [8]

Rotary encoder tersusun dari sebuah piringan tipis yang memiliki lubang pada sekeliling lingkaran. LED ditempatkan pada salah satu sisi piringan sehingga cahaya dapat menembus piringan melalui lubang-lubang tersebut. Di sisi yang lain, sebuah *phototransistor* ditempatkan persis bersebrangan dengan posisi LED. *phototransistor* ini akan menerima cahaya yang dipancarkan oleh LED tersebut dan akan diteruskan ke mikrokontroler sebagai pulsa. Untuk lebih jelasnya, konstruksi dan cara kerja sensor *rotary encoder* dapat dilihat pada Gambar 2.7.



Gambar 2.7 Konstruksi dan Prinsip Kerja Sensor *Rotary Encoder*. [9]

Piringan pada sensor *rotary encoder* dikopel dengan poros motor, sehingga apabila motor berputar, piringan tersebut akan ikut berputar pula. Apabila posisi piringan mengakibatkan cahaya dari LED dapat diterima oleh *phototransistor* melalui lubang-lubang yang ada, maka *phototransistor* akan mengalami saturasi dan menghasilkan suatu pulsa gelombang persegi. Banyaknya deretan pulsa yang dihasilkan *phototransistor* akan menentukan akurasi dari suatu *rotary encoder*. Akibatnya, semakin banyak lubang yang terdapat pada piringan *rotary encoder*, semakin baik pula akurasi dari *rotary encoder* tersebut.

2.4 Arduino Uno

Arduino Uno adalah *board* mikrokontroler berbasis Atmega328. Alat ini mempunyai 14 pin *input*/*output* digital dan 6 diantaranya dapat digunakan sebagai *output* PWM, 6 *input* analog, resonator keramik 19MHz, koneksi USB, soket listrik, ICSP *header,* dan tombol *reset* [1]. Dengan kabel USB alat ini mudah dihubungkan ke komputer dan dapat diaktifkan dengan baterai atau adaptor AC ke DC. Spesifikasi Arduino Uno adalah sebagai berikut :

- Mikrokontroler : Atmega328
  Arduino
- Tegangan operasi : 5 Volt
- Tegangan *input* : 7-12 Volt
  (direkomendasikan)
- Tegangan *input* : 6-20 Volt
  (batasan)

- Pin *input*/*output* digital : 14 (6 diantaranya *output* PWM)
- Pin *input* analog : 6
- Arus DC tiap pin I/O : 40mA
- Arus DC untuk pin 3,3 Volt : 50mA

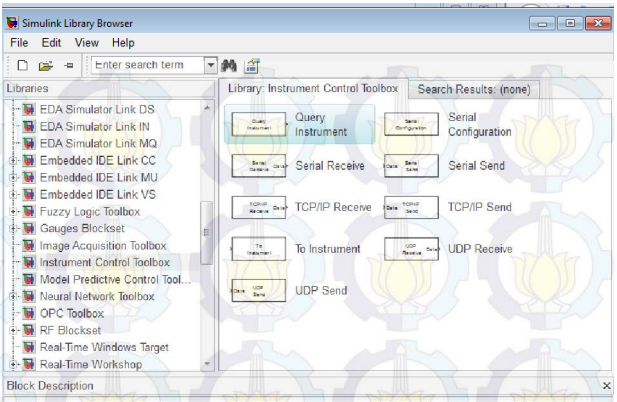
Pada Arduino Uno terdapat 6 *input* analog, dengan nama A0 hingga A5. Pin tersebut memiliki resolusi 10-bit (1024 nilai yang berbeda). *Input* analog ini berupa tegangan dari *ground* ke 5V, walaupun begitu terdapat kemungkinan untuk mengubah batas atas dan bawah dengan menggunakan pin AREF dan fungsi *analogReference()*.

2.5 MATLAB R2014a

MATLAB merupakan paket program dengan bahasa pemrograman yang tinggi untuk mengembangkan algoritma, visualisasi data, dan komputasi numerik. Program MATLAB ini dapat digunakan untuk menyelesaikan masalah komputasi dengan lebih cepat dibandingkan dengan bahasa pemrograman tradisional, seperti C, C++, dan Fortran. MATLAB digunakan untuk banyak aplikasi seperti *signal and image processing*, desain kontrol, pengujian dan pengukuran, permodelan, dan analisis. [1]

Simulink merupakan bagian dari MATLAB untuk memodelkan, mensimulasikan, dan menganalisa sistem dinamik. Simulink dapat membentuk model dari awal atau memodifikasi model yang sudah ada sesuai dengan apa yang diinginkan. Selain itu simulink juga mendukung sistem *linear* dan *non-linear*, permodelan waktu kontinyu atau diskrit, atau gabungan. Simulink ini dapat digunakan sebagai media untuk menyelesaikan masalah dalam industri nyata meliputi kedirgantaraan dan pertahanan, otomotif, komunikasi, elektronik dan pemrosesan sinyal,

Salah satu modul dalam simulink yang dapat digunakan untuk komunikasi perangkat keras adalah *Instrument Control Toolbox*, seperti terlihat pada Gambar 2.8. Modul ini merupakan kunpulan fungsi *m-file* yang dibangun pada lingkungan komputasi teknis MATLAB. *Toolbox* ini menyediakan kerangka kerja untuk komunikasi instrumen yang mendukung GPIB *interface,* standar VISA, TCP/IP, dan protokol UDP. *Toolbox* ini memperluas fitur dasar *serial port* yang ada dalam MATLAB. Selain itu *toolbox* ini berfugsi untuk komunikasi data antara *workspace* MATLAB dan peralatan lainnya. Data tersebut dapat berbentuk biner atau *text*.



Gambar 2.8 Dialog Tampilan *Instrument Control Toolbox.*

Komunikasi *serial* merupakan protokol dasar tingkat rendah untuk komunikasi antara dua peralatan atau lebih. Pada umumnya satu komputer dengan modem, printer, mikrokontroler, atau peralatan lainnya. *Serial port* mengirim dan menerima informasi *bytes* dengan hubungan seri. *bytes* tersebut dikirimkan menggunakan format biner atau karakter ASCII (*American Standard Code for Information Interchange*). Dalam komunikasi serial MATLAB, agar data ASCII dapat diproses *real time*, maka digunakan ASCII *encode* dan *decode* yang terdapat pada *xPC Target Library for RS232.* ASCII *encode* merupakan blok dalam simulink yang digunakan untuk mengubah data *bytes* menjadi karakter ASCII. Sedangkan ASCII *decode* merupakan blok simulink yang digunakan untuk mengubah karakter ASCII menjadi data *bytes* yang kemudian dapat dikonversi sesuai kebutuhan.

2.6 Identifikasi Sistem

Identifikasi sistem merupakan suatu langkah awal dalam menganalisa sistem dinamik. Menurunkan suatu fungsi alih yang baik dan sesuai merupakan salah satu bagian terpenting dalam proses menganalisa sebuah sistem secara keseluruhan. Fungsi alih yang baik dan sesuai cocok digunakan untuk analisa, prediksi, dan desain sistem, *regulator*, dan filter. Fungsi alih memiliki bentuk yang bermacam-macam. Salah satu bentuknya adalah *transfer function* yang cocok untuk permasalahan analisa transien dan sebuah sistem LTI (*Linear Time Invariant*) dan sistem dengan *Single-Input Single-Output* (SISO). Disisi lain, fungsi alih

dengan bentuk *state space* sangat cocok untuk menganalisa suatu sistem dengan *Multiple-Input Multiple-Output* (MIMO).

Fungsi alih dari sebuah sistem dapat diperoleh melalui dua cara yaitu permodelan fisik dan identifikasi sistem. Permodelan fisik merupakan pendekatan analitik berdasarkan hukum fisika seperti hukum newton dan hukum kesetimbangan. Permodelan ini menjelaskan dinamika dalam sistem. Yang kedua adalah identifikasi sistem. Hal ini dilakukan dengan pendekatan eksperimental. Model ini berdasarkan data dari eksperimen yang kemudian didapatkan nilai parameter sistem. Pada beberapa kasus yang sangat kompleks, sangat sulit untuk menentukan model berdasarkan pemahaman fisik. [10]

Identifikasi sistem pada suatu sistem dapat dilakukan dengan menggunakan metode identifikasi statis maupun dinamis. Identifikasi statis dilakukan untuk mendapatkan *gain* dan *time sampling* dari suatu sistem. Identifikasi statis pada suatu sistem dilakukan dengan memberikan suatu *input setpoint* yang bernilai konstan terhadap waktu. Sistem yang digunakan untuk identifikasi statis adalah sistem dengan *loop* terbuka tanpa kontroler. Sedangkan pada identifikasi dinamis, prosedur identifikasinya memiliki beberapa perbedaan mendasar. Diantaranya adalah perbedaan antara sinyal uji dan metode pemodelan yang digunakan. Jika identifikasi statis menggunakan *input setpoint* yang konstan terhadap waktu, maka identifikasi dinamis menggunakan sinyal acak atau *random* sebagai sinyal ujinya. Pada Tugas Akhir kali ini, digunakan metode identifikasi dinamis. Pemilihan metode ini disebabkan karena respon yang sangat cepat dari motor BLDC, sehingga sangat sulit untuk mengamati respon transien dari sistem jika menggunakan identifikasi statis.

2.6.1 Identifikasi Dinamis

Identifikasi dinamis dilakukan untuk mendapatkan pemodelan dari suatu sistem atau *plant*. Identifikasi dinamis memiliki beberapa kelebihan dibandingkan identifikasi statis. Pada identifikasi dinamis, input yang diberikan pada sistem memiliki frekuensi yang berubah-ubah, sehingga karakteristik dan sifat dari sistem dapat diteliti dengan lebih cermat. Input yang diberikan ini dinamakan sinyal *Pseudo-Random Binary Sequence*.

Setelah respon dari sistem diperoleh, langkah selanjutnya adalah melakukan pendekatan fungsi alih dari sistem berdasarkan respon yang didapat. Ada beberapa metode yang bisa dilakukan untuk mendapatkan fungsi alih dari sistem. Salah satunya adalah metode ARX atau *Auto-*

Regressive Exogeneous. Penjelasan selanjutnya akan diberikan pada paragraf dibawah ini.

2.6.1.1 Sinyal Pseudo-Random Binary Sequence

Terdapat beberapa perbedaan mendasar antara identifikasi statis dan dinamis. Diantaranya adalah perbedaan antara sinyal uji dan metode pemodelan yang digunakan. Jika identifikasi statis menggunakan input *setpoint* yang konstan terhadap waktu, maka identifikasi dinamis menggunakan sinyal acak atau *random* sebagai sinyal ujinya. Sinyal ini memiliki frekuensi yang berubah-ubah, sehingga memungkinkan karakteristik sistem dapat diketahui secara lebih teliti. Sinyal tersebut dinamakan sinyal *Pseudo-Random Binary Sequence* (PRBS). Sinyal PRBS mirip dengan bilangan acak secara nyata, tapi juga dapat disebut semu atau *pseudo* karena bersifat deterministik [10]. Sinyal PRBS dapat dihasilkan dari penggunaan *shift register*. Salah satu contoh sinyal uji PRBS dapat dilihat pada Gambar 2.9.

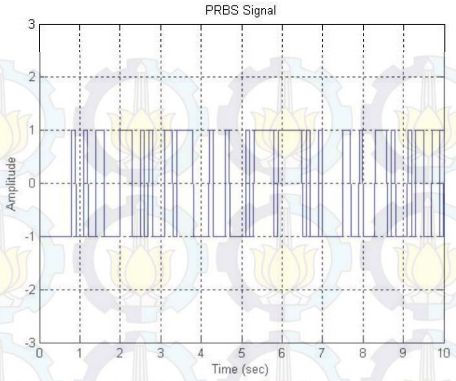
2.6.1.2 Auto-Regressive Exogeneous (ARX)

Langkah selanjutnya adalah menentukan pemodelan yang akan digunakan. Terdapat banyak macam pemodelan yang digunakan pada identifikasi dinamis, antara lain *Auto-Regressive* (AR), *Auto-Regressive Model with External Input* (ARX), *Auto-Regressive Moving Average* (ARMA), dan yang lainnya. Pada pembahasan Tugas Akhir kali ini, digunakan pendekatan ARX untuk mendapatkan fungsi alih yang diharapkan. ARX dipilih karena relatif sederhana dan cukup representatif [10]. Secara matematis, struktur pendekatan ARX dapat dituliskan sebagaimana Persamaan 2.1.

$$y(t)+a_1 y(t-1)+...+a_{n_a} y(k-n_a) = b_1 u(t-n_k)+...+b_{n_b} u(t-n_b-n_k+1)+e(t) \quad (2.1)$$

dengan penjelasan variabel,

$y(t)$	:	*Output* pada waktu *t*
n_a	:	Banyaknya jumlah *pole*
n_b	:	Banyaknya jumlah *zero* ditambah satu
n_k	:	Nilai *dead time*. Jumlah masukan yang terjadi sebelum memberikan pengaruh pada keluaran.



Gambar 2.9 Sinyal Uji *Pseudo-Random Binary Sequence* (PRBS).

Secara ringkas, struktur pemodelan ARX dapat dituliskan sebagaimana Persamaan 2.2

$$A(q)y(t) = B(q)u(t-n_k) + e(t)$$
$$A(q) = 1 + a_1 q^{-1} + \ldots + a_{n_a} q^{-n_a} \quad (2.2)$$
$$B(q) = b_1 + b_2 q^{-1} + \ldots + a_{n_b} q^{n_b+1}$$

2.6.2 Validasi Model

Setelah fungsi alih didapat melalui proses identifikasi sistem, tahapan selanjutnya adalah tahapan validasi model. Validasi model diukur dengan cara mengukur nilai *error* setiap variabelnya. Tujuan dari validasi model adalah untuk mengukur apakah nilai pada pemodelan sudah mendekati dengan nilai sebenarnya dari sistem.

Dalam menggunakan proses validasi model, terdapat beberapa metode dan tolok ukur yang dapat digunakan. Diantaranya adalah metode *Root Mean Square Error* (RMSE). RMSE mengukur akurasi pada nilai deret waktu secara statistik seperti halnya regresi. RMSE dapat merepresentasikan ukuran dari *error* rata-rata karena RMSE membandingkan hasil data pengukuran dan data pemodelan pada skala yang sama antara kedua data tersebut. Formulasi perhitungan RMSE yang mempresentasikan nilai *error* dalam bentuk presentasi dapat dilihat pada Persamaan 2.3 dan Persamaan 2.4.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n} e_i^2}; \quad i = 1,2,3,...,n \tag{2.3}$$

$$e_i = \frac{A_i - M_i}{A_i} \times 100\%; \quad i = 1,2,3,...,n \tag{2.4}$$

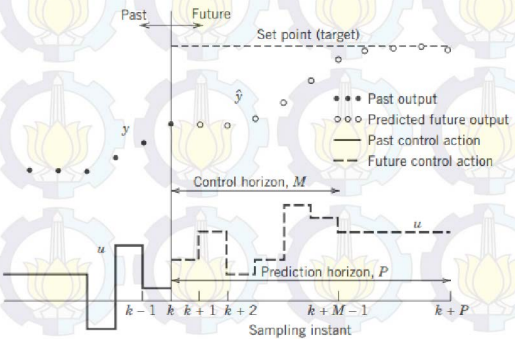
dengan penjelasan variabel,

N	:	Jumlah data
i	:	Urutan data
e_i	:	Nilai *error*
M_i	:	Nilai data hasil pemodelan
A_i	:	Nilai data hasil pengukuran

Semakin kecil nilai pengukuran RMSE suatu pemodelan, semakin baik pula model yang diberikan.

2.7 *Model Predictive Control* (MPC)

Tujuan utama dari sebuah *Model Predictive Control* (MPC) adalah untuk menghitung trayektori dari sinyal kontrol *u* (*manipulated variable*) yang akan datang untuk mengoptimalkan perilaku yang akan datang (*future behavior*) pada sinyal *output y* pada sebuah *plant* berdasarkan pada nilai pengukuran saat ini dan prediksi dari nilai *output* yang akan datang. Objektif dari kontroler MPC adalah untuk menentukan nilai sinyal kontrol (*sequence of control moves*) sehingga nilai *output* yang diprediksi akan mendekati nilai *setpoint* dengan optimal [11]. Konsep umum dari kontroler MPC dapat dilihat pada Gambar 2.10



Gambar 2.10 Konsep dari Kontroler *Model Predictive Control* [12]

Pada Gambar 2.10, dapat dilihat susunan dari nilai *output* saat ini (*actual output*) y, nilai *output* terprediksi (*predicted output*) $\hat{y}$, dan *manipulated input* atau sinyal kontrol u. Pada setiap waktu *sampling* k, kontroler MPC menghitung himpunan dari nilai M atau *control horizon* (selanjutnya disebut N_c) dari *input* $\{u(k+i-1), i=1,2,...,M\}$. Nilai *input* akan ditahan pada nilai konstan setelah M langkah didalam sinyal kontrol tersebut. Nilai *input* akan dihitung sedemikian sehingga nilai himpunan dari P keluaran atau *output* terprediksi $\{y(k+i), i=1,2,...,P\}$ akan mencapai nilai *setpoint* yang diinginkan. P merupakan nilai dari *prediction horizon* (selanjutnya disebut N_p) pada kontroler MPC. Perhitungan nilai kontrol pada kontroler MPC dihitung berdasarkan nilai optimal dari suatu fungsi objektif atau indeks performansi J. [12]

2.7.1 Model *State-Space* dengan *Embedded Integrator*

Pertama, kita mengasumsikan sebuah sistem *single-input single-output* yang dideskripsikan sebagai berikut:

$$\begin{aligned} x_m(k+1) &= A_m x_m(k) + B_m u(k) \\ y(k) &= C_m x_m(k) + \mathrm{D}_m u(k) \end{aligned} \tag{2.5}$$

Dimana u adalah variabel manipulasi atau variabel kontrol, x_m merupakan variabel *state* dan y adalah variabel *output*. Dikarenakan prinsip dari *receding horizon control*, dimana *state* saat ini dibutuhkan untuk menghitung prediksi dan kontrol, maka kita mengasumsikan bahwa *input* $u(k)$ tidak dapat mempengaruhi *output* $y(k)$ pada waktu yang sama. Oleh karena itu, nilai D_m dapat kita abaikan. Oleh karena itu, Persamaan 2.5 dapat ditulis kembali sebagai berikut:

$$\begin{aligned} x_m(k+1) &= A_m x_m(k) + B_m u(k) \\ y(k) &= C_m x_m(k) \end{aligned} \tag{2.6}$$

Untuk dapat menerapkan kontroler MPC pada suatu *plant*, maka kita perlu mengubah bentuk *state space* Persamaan 2.5 menjadi bentuk *augmented model*. Adapun bentuk *augmented model* dapat didefinisikan sebagai berikut:

$$\overbrace{\begin{bmatrix}\Delta x_m(k+1)\\ y(k+1)\end{bmatrix}}^{x(k+1)} = \overbrace{\begin{bmatrix}A_m & 0_m^T\\ C_m A_m & 1\end{bmatrix}}^{A}\overbrace{\begin{bmatrix}\Delta x_m(k)\\ y(k)\end{bmatrix}}^{x(k)} + \overbrace{\begin{bmatrix}B_m\\ C_m B_m\end{bmatrix}}^{B}\Delta u(k)$$
$$y(k) = \overbrace{\begin{bmatrix}0_m & 1\end{bmatrix}}^{C}\begin{bmatrix}\Delta x_m(k+1)\\ y(k+1)\end{bmatrix} \qquad (2.7)$$

Adapun $o_m = \overbrace{\begin{bmatrix}0 & 0 & \cdots & 0\end{bmatrix}}^{n_1}$ dan matriks A,B,C biasa disebut *augmented model*. Model *state space* inilah yang akan digunakan selanjutnya untuk merancang sebuah kontroler MPC.

2.7.2 Perhitungan *Prediction Output* & *Future Control*

Setelah mendapatkan *augmented model*, langkah selanjutnya adalah menghitung nilai output terprediksi dan variabel kontrol yang akan datang. Variabel kontrol yang akan datang dapat ditulis sebagaimana Persamaan 2.8.

$$\Delta u(k_i),\ \Delta u(k_i+1),\ \ldots,\ \Delta u(k_i+N_c-1) \qquad (2.8)$$

N_c merupakan nilai *control horizon*, yaitu jumlah langkah kontrol berkelanjutan yang diterapkan dan diprediksi oleh kontroler MPC dalam sebuah *sampling time.* Selain itu, variabel *output* terprediksi dapat diperkirakan dan diprediksi dalam jumlah sampel N_p, dimana N_p merupakan nilai *prediction horizon*. Adapun variabel *output* terprediksi dapat dituliskan dalam Persamaan 2.9.

$$x(k_i+1|k_i),\quad x(k_i+2|k_i),\ \ldots,\ x(k_i+N_p|k_i) \qquad (2.9)$$

Sebagai catatan, nilai N_c harus lebih kecil atau sama dengan nilai N_p. Setelah itu, nilai *output* terprediksi dan variabel kontrol yang akan datang dapat dihitung dengan menggunakan Persamaan 2.10.

$$Y = Fx(k_i) + \Phi\Delta U \qquad (2.10)$$

Matriks F dan Φ dapat diformulasikan sebagai berikut:

$$F = \begin{bmatrix} CA \\ CA^2 \\ CA^3 \\ \vdots \\ CA^{N_p} \end{bmatrix} ; \Phi = \begin{bmatrix} CB & 0 & \cdots & 0 \\ CAB & CB & \cdots & 0 \\ CA^2B & CAB & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ CA^{N_p-1}B & CA^{N_p-2}B & \cdots & CA^{N_p-N_c}B \end{bmatrix} \quad (2.11)$$

2.7.3 Indeks Performansi Kontroler MPC

Dalam sebuah kontroler MPC, diperlukan proses optimasi yang mempunyai objektif kontrol untuk meminimalkan *error* yang terbentuk dari selisih nilai referensi dengan nilai keluaran dari *plant*. Optimasi tersebut dilakukan dengan mendeskripsikan sebuah nilai dan parameter indeks performansi *J* yang merefleksikan objektif kontrol dari kontroler MPC. Indeks performansi tersebut dapat didefinisikan sebagai berikut:

$$J = (R_s - Y)^T (R_s - Y) + \Delta U^T \overline{R} \Delta U \quad (2.12)$$

$$R_s^T = \overbrace{\begin{bmatrix} 1 & 1 & \cdots & 1 \end{bmatrix}}^{N_p} r(k_i) \quad (2.13)$$

R_s merupakan vektor yang mempunyai informasi sinyal referensi atau *setpoint* yang didefinisikan seperti tertulis pada Persamaan 2.13. Persamaan $(R_s - Y)^T (R_s - Y)$ pada indeks performansi *J* di Persamaan 2.12 mempunyai tujuan untuk meminimalkan *error* yang terjadi antara nilai *predicted output* dengan *setpoint* yang diberikan pada sistem. Sedangkan persamaan $\Delta U^T \overline{R} \Delta U$ merefleksikan seberapa besar nilai ΔU yang akan dihasilkan ketika fungsi objektif indeks performansi *J* dibuat sekecil mungkin. Matriks $\overline{R}$ adalah matriks diagonal yang berbentuk $\overline{R} = r_w I_{N_c \times N_c} \ (r_w \geq 0)$ dan digunakan sebagai parameter *tuning* kontroler MPC. Variabel r_w merupakan *tuning parameter* untuk performa *closed-loop system* pada kontroler MPC. Pada kasus dimana nilai $r_w = 0$, indeks performansi *J* akan mempunyai objektif untuk meminimalkan nilai *error* $(R_s - Y)^T (R_s - Y)$ sekecil mungkin tanpa mempedulikan seberapa besar nilai ΔU yang akan dihasilkan oleh kontroler MPC. Untuk kasus dimana nilai r_w dibuat semakin besar, indeks performansi *J* pada Persamaan 2.12 akan diterjemahkan ke dalam situasi dimana kita akan meminimalkan

nilai *error* $(R_s - Y)^T (R_s - Y)$ secara hati-hati dengan mempertimbangkan seberapa besar nilai ΔU yang akan dihasilkan oleh kontroler MPC.

Untuk mendapatkan nilai kontrol optimal yang akan meminimalisasi indeks performansi *J*, kita dapat mengekspresikan indeks performansi *J* pada Persamaan 2.12 sebagai berikut:

$$\begin{aligned} J &= (R_s - Fx(k_i))^T (R_s - Fx(k_i)) - 2\Delta U^T \Phi^T \\ &\quad (R_s - Fx(k_i)) + \Delta U^T (\Phi^T \Phi + \overline{R}) \Delta U \end{aligned} \tag{2.14}$$

Setelah itu, Persamaan 2.14 diturunkan terhadap ΔU sebagai berikut:

$$\frac{\partial J}{\partial \Delta U} = -2\Phi^T (R_s - Fx(k_i)) + 2(\Phi^T \Phi + \overline{R}) \Delta U \tag{2.15}$$

Kondisi yang dibutuhkan untuk meminimalkan indeks performansi *J* dicari pada kondisi sebagai berikut:

$$\frac{\partial J}{\partial \Delta U} = 0 \tag{2.16}$$

Dari mensubtitusi Persamaan 2.15 dengan Persamaan 2.16, maka kita dapat menyimpulkan solusi optimal dari sinyal kontrol pada kontroler MPC sebagai Persamaan 2.17.

$$\Delta U = (\Phi^T \Phi - \overline{R})^{-1} (\Phi^T \overline{R}_s - \Phi^T Fx(k_i)) \tag{2.17}$$

Nilai $(\Phi^T \Phi - \overline{R})^{-1} \Phi^T R_s$ berhubungan dengan perubahan nilai *setpoint*, sedangkan nilai $-(\Phi^T \Phi - \overline{R})^{-1} \Phi^T F$ berhubungan dengan *state feedbeck control* pada kontroler MPC.

2.7.4 *Closed-loop Control System*

Meskipun nilai optimal parameter pada vektor ΔU mengandung sinyal kontrol $\Delta u(k_i)$, $\Delta u(k_i + 1)$, …, $\Delta u(k_i + N_c - 1)$, kita hanya dapat mengimplementasikan sampel pertama dari urutan atau *sequence* tersebut, contoh $\Delta u(k_i)$, dan mengabaikan nilai atau urutan selanjutnya. Prinsip ini disebut dengan *Receding Horizon Control* (RHC). Ketika periode sampling selanjutnya datang, nilai pengukuran yang paling baru diambil dari *state* vektor $x(k_i+1)$ untuk penghitungan sinyal kontrol yang

baru. Prosedur ini terus berlanjut pada kondisi *real time* untuk memenuhi prinsip RHC. Oleh karena itu, sinyal kontrol yang didapatkan dari Persamaan 2.17 dapat ditulis ulang sebagaimana Persamaan 2.18.

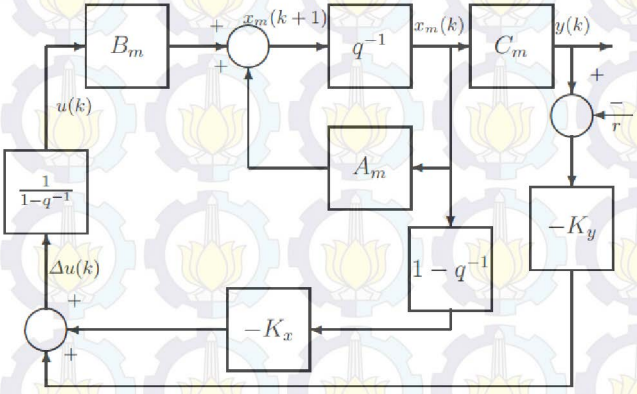
$$\begin{aligned}\Delta u(k_i) &= \overbrace{\begin{bmatrix}1 & 0 & \dots & 0\end{bmatrix}}^{N_c}(\Phi^T\Phi - \overline{R})^{-1}(\Phi^T\overline{R}_s - \Phi^T F x(k_i)) \\ &= K_y r(k_i) - K_{MPC}x(k_i)\end{aligned} \tag{2.18}$$

Nilai K_{MPC} merupakan baris pertama dari $(\Phi^T\Phi + \overline{R})^{-1}(\Phi^T F)$. Sedangkan nilai K_y pada Persamaan 2.18 merupakan baris pertama dari $(\Phi^T\Phi + \overline{R})^{-1}(\Phi^T\overline{R}_s)$. Persamaan 2.18 merupakan bentuk standar dari *linear time-invariant state feedback control*. Nilai *gain* atau penguatan pada *state feedback control* tersebut adalah K_{MPC}.

Untuk mencari persamaan sistem *closed-loop* pada kontroler MPC, digunakan *augmented model* seperti terlihat pada Persamaan 2.19.

$$x(k+1) = Ax(k) + B\Delta u(k) \tag{2.19}$$

Sistem *closed-loop* seperti pada Gambar 2.11 dapat dicari dengan mensubtitusi Persamaan 2.18 kedalam Persamaan 2.19 dan mengganti indeks k_i ke dalam k sebagaimana Persamaan 2.20.



Gambar 2.11 Diagram Blok Sistem Kontrol *Loop* Tertutup *Model Predictive Control* untuk Waktu Diskrit.

$$\begin{aligned} x(k+1) &= Ax(k) - BK_{MPC}x(k) + BK_y r(k) \\ &= (A - BK_{MPC})x(k) + BK_y r(k) \end{aligned} \tag{2.20}$$

Nilai *eigenvalues* pada sistem *closed-loop* diatas dapat ditentukan dengan mengamati persamaan karakteristik *closed-loop* sebagai berikut:

$$\det\left[\lambda I - (A - BK_{MPC})\right] = 0 \tag{2.21}$$

Dikarenakan struktur unik dari matriks *A* dan *C,* kolom terakhir dari matriks *F* identik dengan matriks $\overline{R}_s$, yaitu $\begin{bmatrix} 1 & 1 & \dots & 1 \end{bmatrix}^T$. Oleh karena itu, *gain* K_y identik dengan elemen terakhir pada *gain* K_{MPC}. Perlu dicatat bahwa nilai vektor *state variable* $x(k_i) = \begin{bmatrix} \Delta x_m(k)^T & y(k) \end{bmatrix}^T$ dan dengan definisi K_y, kita dapat mendeklarasikan bahwa *gain* $K_{MPC} = \begin{bmatrix} K_x & K_y \end{bmatrix}$, dimana K_x mendefinsikan hubungan *feedback gain vector* dengan $x_m(k)$. Sedangkan K_y mendefinisikan hubungan antara *feedback gain* dengan $y(k)$. Oleh karena itu, blok diagram sistem *closed-loop* pada kontroler MPC dapat dilihat seperti pada Gambar 2.1. Notasi q^{-1} merupakan notasi *backward shift operator* dan $\frac{1}{1-q^{-1}}$ menotasikan *discrete-time integrator*.

2.7.5 *State Estimation*

Pada perancangan kontroler MPC, kita mengasumsikan bahwa nilai *state* $x(k_i)$ selalu tersedia setiap waktu k_i dan mengasumsikan bahwa semua variabel *state* dapat terukur. Akan tetapi, pada kondisi yang sebenarnya, tidak semua variabel *state* dapat diukur, beberapa diantaranya mustahil untuk diukur. Oleh karena itu, diperlukan sebuah pendekatan untuk mengatasi masalah ini. Salah satu diantaranya adalah dengan mengestimasi variabel $x(k)$ dari pengukuran suatu proses. Sistem yang digunakan untuk mengestimasi variabel yang tidak diketahui ini dinamakan *observer*. Struktur diagram blok untuk kontroler MPC dengan menggunakan *observer* dapat dilihat pada Gambar 2.12.

Pada dasarnya, nilai *state* variabel $x(k_i)$ dapat diestimasi nilainya menggunakan sebuah *observer* dalam bentuk:

$$\hat{x}(k_i+1) = \overbrace{A\hat{x}(k_i) + B\Delta u(k_i)}^{model} + \overbrace{K_{ob}(y(k_i) - C\hat{x}(k_i))}^{correction\ term} \tag{2.22}$$

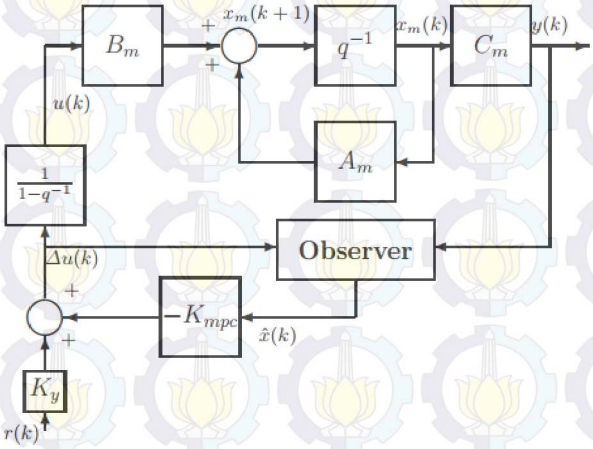
Gain K_{ob} merupakan nilai dari *observer gain matrix*. Pada Persamaan 2.22, terlihat bahwa bagian pertama merupakan *original model* dari sistem, sedangkan bagian kedua merupakan *correction term* yang berdasarkan pada *error* antara nilai *output* yang diukur dengan nilai *output* terprediksi yang menggunakan nilai estimasi $\hat{x}_m(k)$.

Perlu diingat bahwa dalam implementasi kontroler MPC menggunakan *observer*, harus digunakan matriks (A,B,C) dalam bentuk *augmented model*. Dengan mengubah nilai $\hat{x}(k_i)$ menggantikan $x(k_i)$, indeks performansi dapat kita rubah sebagaimana Persamaan 2.23

$$\begin{aligned} J &= (R_s - F\hat{x}(k_i))^T(\overline{R}_s r(k_i) - F\hat{x}(k_i)) + 2\Delta U^T \Phi^T \\ &\quad (R_s - F\hat{x}(k_i)) + \Delta U^T(\Phi^T\Phi + \overline{R})\Delta U \end{aligned} \tag{2.23}$$

Solusi kontrol optimal ΔU dapat diambil menggunakan Persamaan 2.24 sebagai berikut:

$$\Delta U = (\Phi^T\Phi - \overline{R})^{-1}(\Phi^T\overline{R}_s - \Phi^T Fx(k_i)) \tag{2.24}$$



Gambar 2.12 Diagram Blok Sistem Kontrol *Loop* Tertutup *Model Predictive Control* untuk Waktu Diskrit dengan menggunakan *Observer*.

Dengan mengaplikasikan prinsip *Receding Horizon Control* (RHC), dapat kita ambil solusi optimal $\Delta u(k_i)$ pada setiap waktu k_i sebagaimana Persamaan 2.25 berikut.

$$\Delta u(k_i) = K_y r(k_i) - K_{MPC} x(k_i) \qquad (2.25)$$

Persamaan 2.25 merupakan hukum standar *state feedback control* dengan nilai estimasi $x(k_i)$.

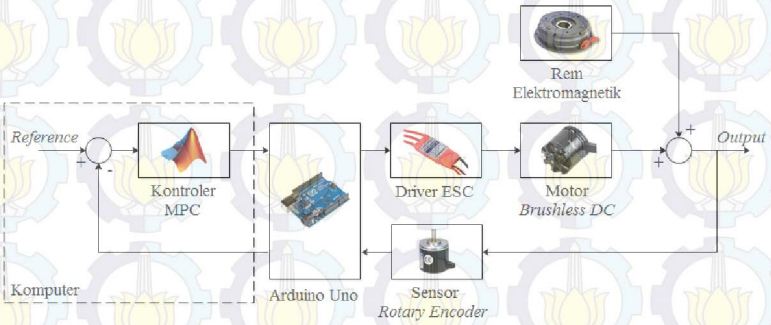
BAB 3
PERANCANGAN SISTEM

Bab 3 pada laporan Tugas Akhir ini akan menjelaskan mengenai perancangan sistem dari pengaturan kecepatan motor BLDC. Selanjutnya, Bab 3 akan menjelaskan mengenai identifikasi dan pemodelan dari sistem. Terakhir, pada bab ini juga akan dijelaskan mengenai perancangan kontroler *Model Predictive Control* pada sistem.

3.1 Gambaran Umum Sistem

Pada pelaksanaan Tugas Akhir kali ini, akan digunakan sebuah *plant* atau sistem yang terdiri dari komponen utama berupa motor *Brushless DC* (BLDC). Pengerjaan *plant* atau sistem ini dikerjakan oleh 5 orang yang kami sebut dengan BLDC Team. *Plant* atau sistem ini kami beri nama BLDC-V1. Selain itu, digunakan rem elektromagnetik untuk memberikan efek pembebanan pada motor BLDC. Pembebanan ini digunakan untuk mensimulasikan kondisi jalan sebenarnya yang akan dilalui oleh kendaraan listrik. Gambar 3.1 menunjukkan blok diagram dari perancangan sistem yang digunakan pada Tugas Akhir kali ini.

Selain perangkat keras berupa motor BLDC dan rem elektromagnetik, ditambahkan pula beberapa komponen pendukung, seperti *driver* untuk motor BLDC dan rem elektromagnetik. Untuk komponen pengukuran kecepatan, digunakan sensor *rotary encoder* yang digunakan sebagai umpan balik yang akan diteruskan ke dalam komputer.



Gambar 3.1 Blok Diagram Sistem Pengaturan Kecepatan Motor *Brushless DC* (BLDC).

Dalam sistem ini, motor BLDC merupakan komponen yang dikontrol untuk mencapai *output* yang diinginkan. Demi tujuan tersebut, digunakan kontroler berbasis *Model Predictive Control* (MPC) guna mencapai performansi yang diinginkan. Kontroler ini akan mengeluarkan sinyal kontrol yang kemudian dikirimkan ke *driver* untuk menggerakkan motor BLDC. Dengan begitu, *output* yang dihasilkan diharapkan bisa sesuai dengan referensi yang diinginkan.

3.2 Perancangan Perangkat Keras

Pada tahap perancangan keras, terdapat 2 jenis perancangan yang dilakukan, yaitu perancangan mekanik dan elektronik. Perancangan mekanik merupakan perancangan komponen utama pada *plant*, yaitu berupa motor BLDC dan rem elektromagnetik. Sedangkan perancangan elektronik merupakan perancangan untuk kontroler, *driver* dan rangkaian sensor yang akan digunakan pada *plant* ini. Kontroler akan diprogram didalam PC dan sebagai perantara komputer dengan *plant*, digunakan mikrokontroler Arduino yang juga berfungsi sebagai perangkat akuisisi data.

Arduino akan menerima data dari sensor dan mengirimkannya kepada komputer. Selain itu, digunakan digunakan pula rangkaian *driver* untuk menggerakan motor BLDC dan memberi *input* arus pada rem elektromagnetik. Terdapat pula rangkaian sensor yang berfungsi mengukur *input* arus yang masuk ke dalam rem elektromagnetik dan sensor kecepatan motor BLDC berupa sensor *rotary encoder*.

3.2.1 Perancangan Mekanik

Pada *plant* sistem pengaturan kecepatan motor BLDC, terdapat beberapa komponen utama yang digunakan, antara lain motor BLDC sebagai tenaga penggerak dan objek yang dikontrol. Selain itu, terdapat rem elektromagnetik yang digunakan untuk memberikan efek pembebanan pada motor BLDC. Rem elektromagnetik diberikan *input* arus DC yang berbentuk PWM (*Pulse Width Modulation*). Untuk lebih jelasnya, penjelasan mengenai konstruksi motor BLDC dan rem elektromagnetik dapat dilihat pada subbab dibawah ini.

3.2.1.1 Motor Brushless DC

Motor BLDC yang digunakan merupakan motor BLDC yang digunakan pada pesawat *aeromodelling*. Motor BLDC jenis ini merupakan miniatur dari motor BLDC penggerak kendaraan listrik

dikarenakan konstruksinya yang lebih kecil. Motor yang digunakan merupakan berasal dari produsen RCTimer dengan nomor seri HP2212-1000KV. Bentuk fisik motor BLDC yang digunakan pada *plant* ini dapat dilihat pada Gambar 3.2. Sedangkan spesifikasi motor BLDC dapat dilihat pada Tabel 3.1.

3.2.1.2 Electronic Speed Control (ESC)

Electronic Speed Control atau yang biasa disingkat ESC merupakan *driver* atau *inverter* yang mengubah arus DC dalam bentuk PWM menjadi tegangan AC yang dapat digunakan untuk menggerakkan motor BLDC. ESC biasa digunakan pada *quadcopter* untuk menggerakkan motor BLDC.

Tabel 3.1 Spesifikasi Motor BLDC RCTimer HP2212-1000KV. [13]

Parameter	Nilai	
Kekuatan Magnet	40 µH	
Berat Motor	59,06 gram	
KV	1002,7 RPM/Volt	
Kecepatan Motor	Tanpa Beban	11.130 RPM
	Beban Maksimal	6.840 RPM
Input Arus	Tanpa Beban	0,7 Ampere
	Beban Maksimal	13,9 Ampere
Power Motor	Beban Minimal	44,4 Watt
	Beban Maksimal	154,29 Watt
Input Baterai	Lithium Polimer 2S-4S	

Gambar 3.2 Motor *Brushless* DC Tipe *Outrunner* dengan Tipe RCTimer HP2212-1000KV. [13]

Input yang diberikan sebagai masukan ke dalam ESC ini sebesar 20 ms atau 50 Hz. Untuk kecepatan minimal, nilai sinyal PWM yang masuk ke dalam ESC bernilai 1 ms. Sedangkan untuk kecepatan maskimal, dibutuhkan sinyal PWM sebesar 2 ms untuk kecepatan hingga 13.000 RPM. Sinyal PWM ini akan diolah oleh *timer* dan *mikrokontroler* yang ada pada ESC untuk mengaktifkan rangkaian *power transistor* yang terdiri dari beberapa MOSFET. Rangkaian *power transistor* inilah yang memberikan *input* arus ke dalam motor BLDC berupa sinyal sinusoidal 3 fasa dengan perbedaan sudut sebesar 120°.

Pada Tugas Akhir kali ini, digunakan ESC Turnigy PLUSH-25A. ESC ini sanggup memberi arus pada motor BLDC hingga 25 Ampere. Untuk lebih jelasnya, spesifikasi dan bentuk dari ESC Turnigy PLUSH-25A dapat dilihat pada Tabel 3.2 dan Gambar 3.3 berikut.

Gambar 3.3 *Electronic Speed Control* Turnigy PLUSH-25A. [14]

Tabel 3.2 Spesifikasi ESC Turnigy PLUSH-25A. [14]

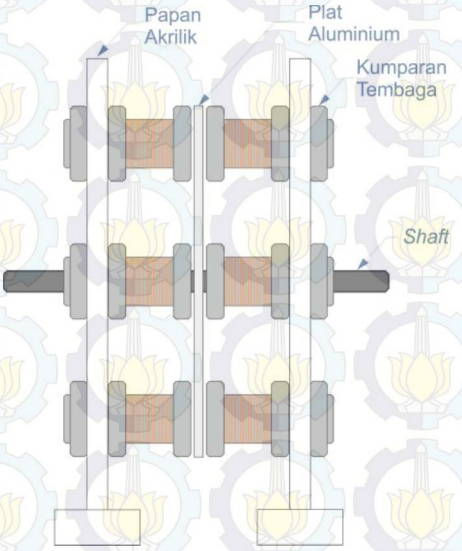
Parameter	Nilai	
Model	Turnigy PLUSH-25A	
Continous Current	25 Ampere	
Burst Current (>10s)	35 Ampere	
Kecepatan Motor	2 *pole*	210.000 RPM
	6 *pole*	70.000 RPM
	12 *pole*	35.000 RPM
Input Baterai	Lithium Polimer 2S-4S	
Dimensi	22 gram	
Berat	45 mm x 24 mm x 11 mm	

3.2.1.3 Rem Elektromagnetik

Alat ini digunakan untuk memberikan efek pembebanan pada motor BLDC. Bentuk perancangan rem elektromagnetik yang digunakan pada *plant* ini dapat dilihat pada Gambar 3.4. Medan elektromagnetik dari rem ini dihasilkan oleh beberapa kumparan yang dihubungkan secara seri dan diberikan masukan arus DC. Arus DC yang masuk ke dalam rem elektromagnetik berbentuk PWM yang *duty cycle*-nya diatur oleh *driver* dan Arduino. Adapun sumber tegangannya didapat dari jala jala PLN yang disearahkan melalui rangkaian *rectifier*.

Rem elektromagnetik ini disusun dari 8 kumparan yang disusun secara seri. 8 kumparan ini dipisahkan menjadi 2 bagian dan diantara celah tersebut dipasang piringan aluminium. Piringan aluminium tersebut lalu dipasang ke dalam sebuah *shaft* yang selanjutnya dikopel dengan motor BLDC. Perbandingan *gear* yang digunakan untuk mengkopel motor BLDC dengan shaft tersebut adalah 1:4.

Konstruksi rem sendiri dari 4 kumparan pada tiap sisinya. Tegangan input yang diberikan pada tiap sisi sebesar 36 Volt. Jumlah lilitan tiap kumparan dihitung berdasarkan rumus berikut ini:



Gambar 3.4 Perancangan Konstruksi Fisik Rem Elektromagnetik.

$$N = \frac{44}{d} \times V = \frac{44}{1,4} \times 36 = 1131 \qquad (3.1)$$

N : Jumlah lilitan tiap sisi
d : Diameter kumparan (cm)
V : Tegangan input (Volt)

Jumlah tersebut masih harus dibagi 4 dikarenakan tiap sisi rem yang mempunyai 4 kumparan. Jadi, tiap kumparan memiliki 283 lilitan.

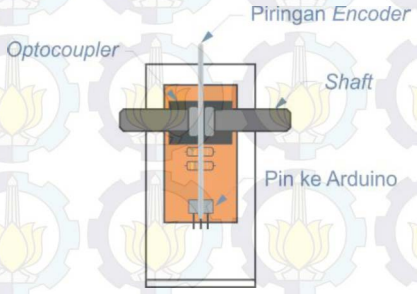
3.2.2 Perancangan Elektronik

Perancangan elektronik ini meliputi desain *layout* rangkaian PCB serta pengkabelan. Rangkaian elektronik pada *plant* ini meliputi rangkaian *driver* rem elektromagnetik, rangkaian penyearah gelombang AC atau *rectifier* serta rangkaian sensor kecepatan *rotary encoder* dan sensor arus. Selain itu, digambarkan pula rancangan pengkabelan pada mikrokontroler Arduino.

3.2.2.1 Rangkaian Sensor Rotary Encoder

Rotary encoder merupakan salah satu jenis sensor yang biasa digunakan untuk mengukur kecepatan. Sensor ini dipasang pada *shaft* seperti yang terlihat seperti Gambar 3.5. Sensor terdiri dari piringan besi yang mempunyai lubang dengan jumlah dan sudut yang tertentu.

Sensor *rotary encoder* terdiri dari rangkaian *optocoupler* yang tersusun atas *Light Emitting Diode* atau (LED) dan *receiver phototransistor* seperti terlihat pada Gambar 3.6. Apabila *optocoupler* tidak terhalang apapun, LED akan memancarkan cahaya dan akan diterima oleh *phototransistor* sehingga menghasilkan pulsa.

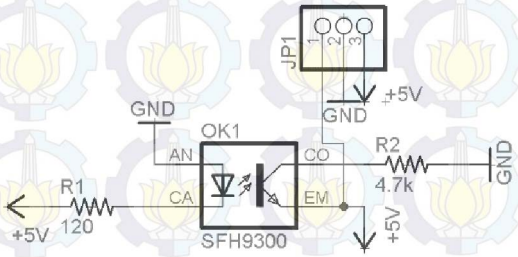


Gambar 3.5 Sensor *Rotary Encoder* yang Terpasang pada *Shaft*.

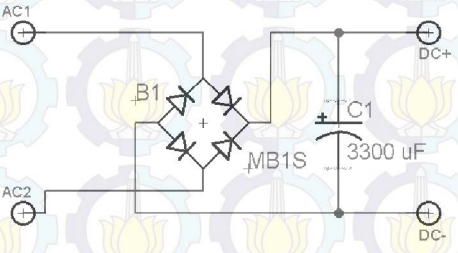
Tetapi, ketika *optocoupler* terhalang, cahaya dari LED tidak bisa diterima oleh *phototransistor* sehingga tidak menghasilkan pulsa apapun. Untuk lebih jelasnya, rangkaian penyusun sensor *rotary encoder* ini dapat dilihat pada Gambar 3.6

3.2.2.2 Rangkaian Penyearah Gelombang AC

Untuk menjalankan komponen rem elektromagnetik, dibutuhan suatu tegangan DC yang dialirkan pada kumparan sehingga menghasilkan medan elektromagnetik. Oleh karena itu, diperlukan suatu rangkaian yang mengubah tegangan AC yang berasal dari jala-jala listrik PLN dan mengubahnya menjadi tegangan DC. Rangkaian tersebut dinamakan rangkaian penyearah gelombang AC atau yang biasa dikenal dengan nama rangkaian *rectifier*. Bentuk skematik dari rangkaian *rectifier* dapat dilihat pada Gambar 3.7. Rangkaian *rectifier* tediri dari 4 buah *dioda* (*diode bridge*) yang disusun seperti pada Gambar 3.7.



Gambar 3.6 Skema Rangkaian *Rotary Encoder* yang Tersusun dari *Optocoupler* dan Resistor.

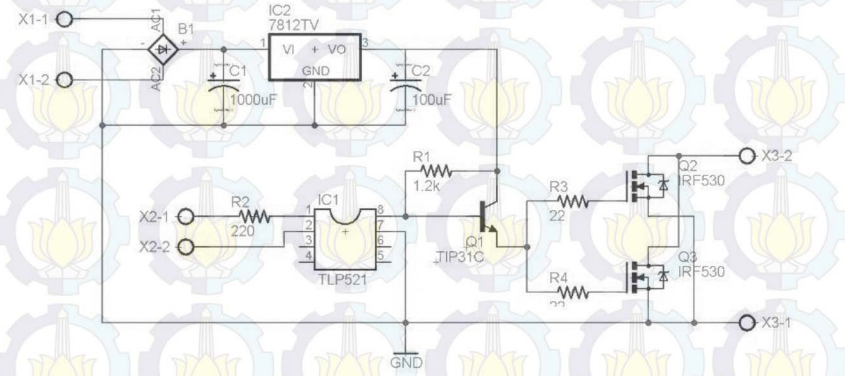


Gambar 3.7 Rangkaian Penyearah Gelombang AC yang Terdiri dari *Diode Bridge* dan Kapasitor.

Sifat dari dioda yang unik, yaitu *short circuit* ketika *forward bias* dan *open circuit* ketika *reverse bias* dimanfaatkan untuk mengubah tegangan AC menjadi tegangan DC. Setelah keluar dari *diode bridge* tegangan DC tidak dapat langsung dialirkan ke dalam kumparan. Ini dikarenakan tegangan DC tersebut masih memiliki tegangan riak atau *ripple* yang dapat merusak komponen. Untuk meminimalisir riak tersebut, digunakanlah sebuah kapasitor yang berfungsi sebagai filter dan menghilangkan tegangan *ripple* tersebut.

3.2.2.3 Rangkaian Driver Rem Elektromagnetik

Rem elektromagnetik sangat berperan penting dalam kesulurahan sistem ini. Rem digunakan untuk memberikan pembebanan pada motor BLDC. Untuk mengatur nilai pembebanan yang akan diberikan kepada motor BLDC, dibutuhkan suatu rangkaian yang digunakan untuk mengatur besaran arus DC yang masuk ke dalam kumparan. Oleh karena itu, dibuatlah suatu *driver* yang digunakan untuk mengatur arus yang masuk ke dalam kumparan. Arus ini berupa arus DC yang telah disearahkan oleh rangkaian penyearah. Pengaturan arus yang masuk ke dalam kumparan tersebut dilakukan menggunakan teknik PWM atau *Pulse Width Modulation* yang diatur menggunakan mikrokontroler Arduino. Rangkaian skematik *driver* rem elektromagnetik dapat dilihat pada Gambar 3.8. Sumber tegangan pada rangkaian driver ini berasal dari tegangan AC.

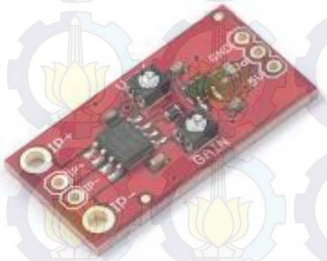


Gambar 3.8 Skema Rangkaian *Driver* Rem Elektromagnetik.

Pada prinsipnya, *driver* rem ini mempunyai kemiripan dengan *driver* yang biasa digunakan untuk menjalankan motor DC. *Driver* ini berkerja dengan diberikan tegangan 12V untuk mengaktifkan komponen komponen seperti regulator (IC7812), *transistor* (TIP31C), dan *optocoupler* (TLP521). Cara kerjanya, ketika mikrokontroler memberikan tegangan dengan *duty cycle low stage*, maka terdapat beda potensial antara mikrokontroler dengan *regulator* sehingga *optocoupler* akan aktif. Saat itu, *transistor* tidak akan aktif karena tegangan dialirkan oleh *transistor* yang ada pada *optocoupler*. Saat mikrokontoler memberikan tegangan dengan *duty cycle high stage*, tidak ada beda potensial antara *regulator* dan mikrokontroler sehingga *optocoupler* tidak aktif. Karena itu tegangan 12 V akan mengalir melalui basis dari *transistor* dan mengaktifkan *transistor* tersebut. Tegangan dari kolektor mengalir ke emitor dan akan mengaktifkan basis dari dua MOSFET (IRF530) yang dipasang paralel. Basis pada kedua MOSFET aktif, maka tegangan supply akan dapat mengalir ke kedua MOSFET tersebut dan mengaktifkan rem elektromagnetik.

3.2.2.4 Rangkaian Sensor Arus

Sensor arus digunakan untuk mengukur besaran nilai arus yang dialirkan ke dalam rem elektromagnetik. Besaran arus ini sangat penting untuk diketahui sebab akan digunakan sebagai parameter dan nilai pembebanan yang akan diberikan kepada motor. Sensor arus yang digunakan berjenis sensor ACS712 buatan produsen SparkFun. Sensor ini dapat mengukur arus dengan batas maksimal mencapai 5 Ampere. Tabel 3.3 akan menjelaskan dengan sekilas mengenai spesifikasi dari sensor arus ini. Gambar dari sensor arus yang digunakan pada Tugas Akhir ini dapat dilihat pada Gambar 3.9.



Gambar 3.9 Sensor Arus SparkFun ACS712 5A Breakout

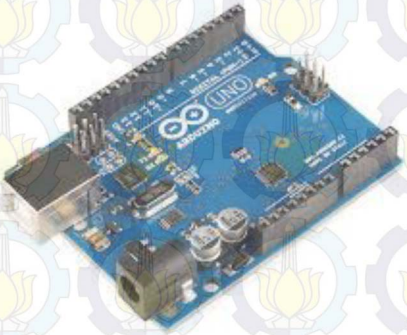
Tabel 3.3 Spesifikasi Sensor Arus SparkFun ACS712 5A. [15]

Parameter	Nilai
Model	SparkFun ACS712 5A Breakout
Arus Maksimal	5 Ampere
Bandwith	80 kHz
Sensitivitas *Output*	66-185 mV/A
Tegangan Operasi	5 Volt
Tegangan *Output*	2,5-5 Volt

3.2.2.5 Mikrokontroler Arduino

Pada *plant* kali ini, digunakan mikrokontroler Arduino sebagai alat akuisisi data yang menjembatani antara *software* dan *aktuator* beserta sensor. Arduino digunakan karena pemrogramannya yang terbilang sederhana dan cukup fleksibel. Adapun jenis Arduino yang digunakan pada pelaksanaan Tugas Akhir kali ini adalah jenis Arduino Uno R3 seperti terlihat pada Gambar 3.10

Arduino Uno memiliki 14 pin *input/output* yang mana 6 pin dapat digunakan sebagai *output* PWM, 6 analog *input*, *crystal* osilator 16 MHz, koneksi USB, *jack power*, kepala ICSP, dan tombol *reset*. Pengiriman data dari sistem minimum Arduino ke PC/laptop dilakukan melalui koneksi USB. Bahasa yang digunakan oleh perangkat Arduino ini yaitu menggunakan bahasa C+ untuk pemrogramannya. *Software* Arduino ini dilengkapi dengan kumpulan *library* yang cukup lengkap, sehingga dapat membantu pengguna dalam penggunaannya. Sistem minimum Arduino Uno R3 secara lengkap dapat dilihat pada Gambar 3.10.



Gambar 3.10 Mikrokontroler Arduino Uno R3. [16]

3.3 Perancangan Perangkat Lunak

Perangkat lunak diperlukan dalam perancangan sistem sebagai *interface* antara *plant* dan komputer. Perangkat lunak yang digunakan dalam pengerjaan Tugas Akhir ini antara lain adalah penggunaan perangkat lunak Arduino dan MATLAB. Arduino digunakan sebagai alat akuisisi data, sedangkan MATLAB digunakan untuk membaca nilai dari sensor untuk keperluan identifikasi sistem dan mengirimkan sinyal kontrol pada ESC. Selain itu, MATLAB digunakan pula sebagai *software* untuk desain, simulasi dan implementasi kontroler.

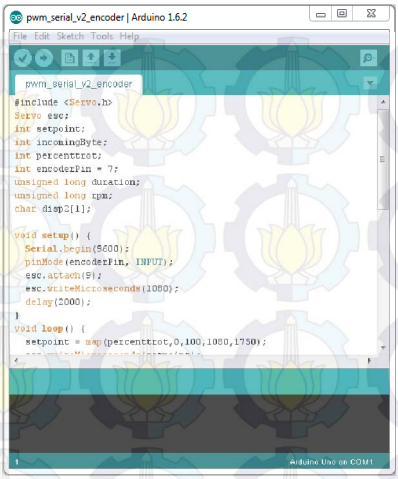
3.3.1 *Software* Arduino

Seperti telah yang dijelaskan sebelumnya, mikrokontroler Arduino digunakan sebagai alat akuisisi data dari berbagai macam komponen pada *plant*. *Software* yang sering digunakan untuk meng-*compile* bahasa pemrograman pada Arduino bernama Arduino IDE. Program akuisisi data yang dikehendaki ditulis pada Arduino IDE di komputer, lalu di-*compile* dan di-*upload* menuju mikrokontroler via serial USB. Dalam Tugas Akhir kali ini, mikrokontroler Arduino secara garis besar digunakan untuk menerima data kecepatan dari sensor *rotary encoder* melalui pin 7, mengendalikan besaran PWM pada *driver* rem elektromagnetik melalui pin A0, menerima data dari sensor arus pada pin A2 dan mengirimkan *throttle* atau sinyal kontrol berupa PWM pada ESC pada pin 9. Tampilan program Arduino IDE dapat dilihat pada Gambar 3.11.

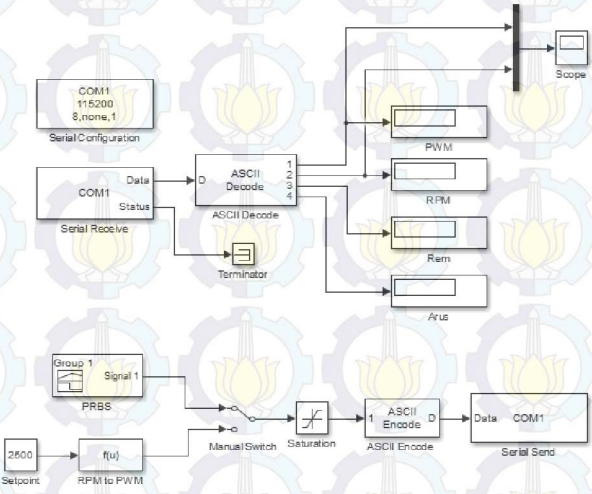
3.3.2 Software MATLAB

Selain Arduino IDE, *software* lain yang digunakan pada Tugas Akhir kali ini adalah *software* MATLAB. Versi MATLAB yang digunakan pada pelaksanaan Tugas Akhir kali ini mempunyai versi 2015a. *Software* MATLAB merupakan *software* yang sangat vital dan umum digunakan untuk desain kontroler beserta implementasinya. Pada *software* MATLAB, terdapat sub-program lainnya yang bernama Simulink. *Software* Simulink digunakan sebagai *Human Machine Interface* pada proses pengiriman dan penerimaan data melalui serial USB dari mikrokontroler Arduino seperti terlihat pada Gambar 3.12.

Selain itu, pada *software* Simulink juga terdapat blok-blok yang dapat digunakan untuk mendesain dan mengimplementasikan kontroler. Disisi lain, Simulink juga digunakan untuk proses identifikasi sistem *open loop* maupun *closed loop* menggunakan blok *Instrument Control Toolbox* melalui blok *Serial Send* dan *Serial Receive*.



Gambar 3.11 Tampilan Arduino IDE



Gambar 3.12 Blok Identifikasi Sistem *Open Loop* pada Simulink.

3.4 Identifikasi dan Pemodelan Sistem

Identifikasi merupakan sebuah proses yang harus dilewati guna mendapatkan parameter dan bentuk matematika dari sebuah sistem. Pada proses ini, identifikasi sistem dilakukan dengan memberikan beberapa nilai pembebanan yang diberikan pada motor BLDC. Setelah didapatkan respon dalam beberapa parameter pembebanan, respon tersebut dapat dicari fungsi alihnya melalui identifikasi dinamis.

3.4.1 Metode Pembebanan *Plant*

Pada Tugas Akhir kali ini, motor BLDC diberikan beban berupa rem elektromagnetik. Beban ini menggambarkan beban awal pada kendaraan elektrik yang ditenagai oleh motor BLDC. Pembebanan dilakukan dengan 3 nilai yang berbeda, yaitu beban minimal, nominal dan maksimal. Potensiometer digunakan untuk mengatur pemberian *input* PWM ke *driver* rem elektromagnetik melalui pin A0. Skala yang digunakan mulai dari nol untuk menon-aktifkan rem hingga skala 1000 untuk pemberian *input* maksimal pada rem. Nilai tegangan yang masuk ke dalam rem elektromagnetik dapat dilihat pada Tabel 3.4.

3.4.2 Metode Identifikasi dan Pemodelan

Identifikasi pada motor BLDC dilakukan pada sistem *open loop* menggunakan identifikasi dinamis. Identifikasi dinamis dilakukan dengan memberikan sinyal acak atau *random*. Sinyal ini biasa disebut dengan sinyal PRBS (*Pseudo-Random Binary Sequence*). Adapun input yang diberikan berupa nilai pulsa PWM yang dimasukkan ke dalam ESC. Pengambilan data dilakukan dengan membaca nilai sensor *rotary encoder* yang telah dihubungkan ke dalam Arduino. Mikrokontroler Arduino lalu dihubungkan ke dalam komputer melalui komunikasi *serial* untuk dibaca nilai *input* PWM dan *output* kecepatannya.

Setelah respon berhasil didapatkan, respon tersebut lalu dimodelan dengan pendekatan ARX pada orde 1 sampai dengan 3 menggunakan bantuan *System Identification Toolbox* pada *software* MATLAB. Lalu, model yang digunakan adalah model dengan nilai *error* yang paling kecil.

Tabel 3.4 Metode Pembebanan pada Sistem.

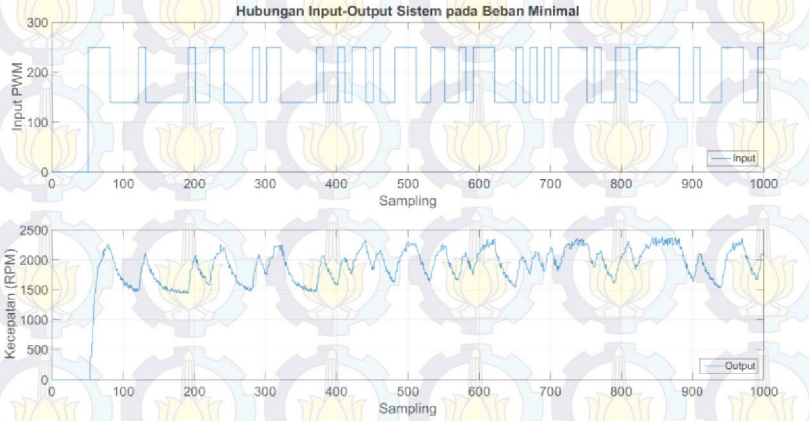
Metode Pembebanan	*Input* PWM	Nilai
Minimal	400	4,4 Volt
Nominal	600	7,89 Volt
Maksimal	800	12,56 Volt

3.4.3 Pemodelan Motor *Brushless* DC

Pemodelan motor BLDC didapatkan dari identifikasi dinamis melalui sistem *open loop*. Dalam hal ini dilakukan 3 jenis pemodelan, yaitu pemodelan beban minimal, nominal dan maksimal. Data yang telah didapatkan kemudian dianalisis menggunakan bantuan *System Identification Toolbox* pada *software* MATLAB untuk mendapatkan fungsi alih sistem melalui pendekatan ARX. Untuk memperjelas, Gambar 3.13 menyajikan perbandingan data mengenai hubungan *input-output* sistem identifikasi dinamis.

3.4.3.1 Beban Minimal

Pada beban minimal, motor BLDC diberikan pembebanan rem elektromagnetik pada tegangan 4,4 Volt. Setelah mengetahui respon dari sistem, barulah dilakukan pemodelan pada sistem melalui pendekatan ARX. Pemodelan sistem dilakukan hingga orde 3. Identifikasi dinamis untuk pemodelan ARX pada orde 1 sampai orde 3 pada beban minimaldapat diamti pada Tabel 3.5 berikut ini.

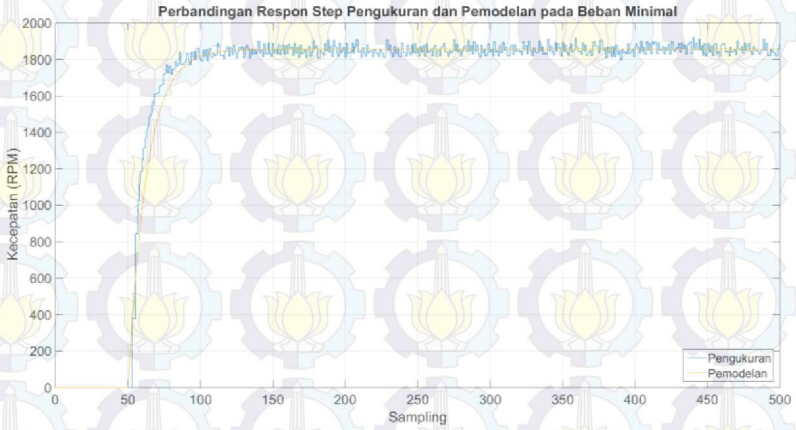


Gambar 3.13 Hubungan *Input-Output* Sistem pada Identifikasi Dinamis.

Gambar 3.14 menunjukkan perbandingan antara hasil respon pengukuran dengan respon pemodelan saat sistem diberikan sinyal masukan berupa sinyal *unit step*.

Tabel 3.5 Identifikasi Dinamis pada Beban Minimal.

Orde	Fungsi alih	RMSE (%)
1	$\frac{0,007196}{z-0,9992}$	11,7
2	$\frac{0,002513z+0,004715}{z^2-0,9957z-0,003521}$	6,98
3	$\frac{0,002525z^2-6,218\times10^{-15}z+0.004736}{z^3-0,9957z^2-1,846\times10^{-15}z-0.003537}$	7,38



Gambar 3.14 Perbandingan Respon Hasil Pengukuran dan Pemodelan pada Beban Minimal

3.4.3.2 Beban Nominal

Pada beban nominal, motor BLDC diberikan beban berupa rem elektromagnetik pada tegangan 7,77 Volt. Setelah mengetahui respon dari sistem, barulah dilakukan pemodelan pada sistem melalui pendekatan ARX. Pemodelan sistem dilakukan hingga orde 3. Identifikasi dinamis untuk pemodelan ARX pada orde 1 sampai orde 3 pada beban nominal dapat diamti pada Tabel 3.6 berikut ini. Identifikasi dinamis untuk pemodelan ARX pada orde 1 sampai orde 3 pada beban nominal dapat diamati pada Tabel 3.6 berikut ini.

Tabel 3.6 Identifikasi Dinamis pada Beban Nominal.

Orde	Fungsi alih	RMSE (%)
1	$\frac{0,007487}{z-0,9991}$	9,59
2	$\frac{0,003239z+0,004284}{z^2-0,9956z-0,003566}$	5,68
3	$\frac{0,003254z^2+9,887\times10^{-17}z+0.004303}{z^3-0,9955z^2-7,971\times10^{-15}z-0.003583}$	7,96

3.4.3.3 Beban Maksimal

Pada pembebanan terakhir, yaitu pada kondisi beban maksimal, motor akan diberikan beban dari rem elektromagnetik yang diberi input tegangan sebesar 12,56 Volt. Setelah mengetahui respon dari sistem, barulah dilakukan pemodelan pada sistem melalui pendekatan ARX. Pemodelan sistem dilakukan hingga orde 3. Identifikasi dinamis untuk pemodelan ARX pada orde 1 sampai orde 3 pada beban maksimal dapat diamati pada Tabel 3.7 berikut ini.

Tabel 3.7 Identifikasi Dinamis pada Beban Maksimal.

Orde	Fungsi alih	RMSE (%)
1	$\frac{0,009326}{z-0,9988}$	5,89
2	$\frac{0,002823z+0,006564}{z^2-0,9936z-0,005205}$	5,43
3	$\frac{0,002843z^2-2,475\times10^{-15}z+0.006606}{z^3-0,9935z^2-6,35\times10^{-15}z-0.005239}$	8,19

3.4.4 Pengujian dan Validasi

Setelah dilakukan pemodelan menggunakan pendekatan ARX, langkah selanjutnya adalah memvalidasi fungsi alih tersebut. Semakin baik suatu pemodelan, maka respon yang dihasilkan akan semakin menyerupai respon aslinya. Keakuratan suatu pemodelan dapat dilihat dari nilai *error*-nya, dalam hal ini dikalkulasi menggunakan metode *Root Mean Square Error* (RMSE). Semakin kecil nilai RMSE, semakin baik pula pemodelan yang dihasilkan.

Tabel 3.8 Pemodelan Motor BLDC pada Berbagai Macam Kondisi Pembebanan

Pembebanan	Fungsi alih	RMSE (%)
Minimal	$\frac{0,002513z+0,004715}{z^2-0,9957z-0,003521}$	6,98
Nominal	$\frac{0,003239z+0,004284}{z^2-0,9956z-0,003566}$	5,68
Maksimal	$\frac{0,002823z+0,006564}{z^2-0,9936z-0,005205}$	5,43

Berdasarkan hasil pemodelan diatas, kita dapat menyimpulkan bahwa pemodelan ARX dengan orde 2 mempunyai nilai *error* yang paling rendah untuk semua kondisi pembebanan. Oleh karena itu, pemlihan fungsi alih pada semua kondisi pembebanan yang akan digunakan pada simulasi dan implementasi dapat dilihat pada Tabel 3.8.

3.5 Perancangan Kontroler *Model Predictive Control*

Setelah fungsi alih dari sistem telah didapatkan, langkah selanjutnya adalah merancang kontroler untuk masing-masing pembebanan. Kontroler digunakan untuk mengembalikan respon ke nilai *setpoint* yang diinginkan sekalipun motor BLDC diberi beban. Pada perancangan kontroler ini, akan digunakan fungsi alih sistem pada pembebanan maksimal dikarenakan fungsi alih ini memiliki nilai RMSE paling kecil. Tahapan desain kontroler ini meliputi perancangan fungsi alih pada model *state-space*, desain *augmented model*, perancangan *state estimator* pada sistem menggunakan *observer* dan penentuan parameter kontroler *Model Predictive Control* (MPC).

3.5.1 Perancangan Model *State Space*

Kontroler MPC merupakan kontroler berbasis model. Artinya, diperlukan sebuah pemodelan fungsi alih yang baik agar kontroler yang telah didesain dapat bekerja secara optimal. Fungsi alih yang digunakan merupakan fungsi alih motor BLDC pada pembebanan maksimal. Fungsi alih ini tidak bisa langsung digunakan untuk merancang kontroler MPC, melainkan harus diubah terlebih dahulu ke dalam bentuk *state-space*.

Pada pembebanan maksimal, representasi fungsi alih yang bernilai RMSE paling kecil adalah sebagai berikut:

$$\frac{Y(z)}{U(z)} = \frac{b_0 z^2 + b_1 z + b_2}{z^2 + a_1 z + a_2} = \frac{0,002823z + 0,006564}{z^2 - 0,9936z - 0,005205} \quad (3.2)$$

Berdasarkan fungsi alih pada Persamaan 3.2, kita dapat menyimpulkan variabel-variabel sebagai berikut:

$a_1 = -0,9936$

$a_2 = -0,005205$

$b_0 = 0$

$b_1 = 0,002823$

$b_2 = 0,006564$

Bentuk *state-space* yang digunakan pada Tugas Akhir ini adalah bentuk *controllable canonical form* [17]. Bentuk *state-space* untuk orde 2 tersebut mempunyai struktur sebagai berikut:

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -a_2 & -a_1 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(k)$$

$$y(k) = \begin{bmatrix} b_2 - a_2 b_0 & b_1 - a_1 b_0 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + b_0 u(k) \quad (3.3)$$

Berdasarkan bentuk diatas, maka model *state-space* dari Persamaan 3.1 dapat kita simpulkan sebagai berikut:

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0,005205 & 0,9936 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(k)$$

$$y(k) = \begin{bmatrix} 0,006564 & 0,002823 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix}$$

3.5.2 Desain *Augmented Model*

Selanjutnya adalah mengubah bentuk *state-space* dari sistem ke dalam bentuk *augmented model* dari model diatas sebagai berikut:

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \end{bmatrix} = \overbrace{\begin{bmatrix} 0 & 1 \\ 0,005205 & 0,9936 \end{bmatrix}}^{A_m} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + \overbrace{\begin{bmatrix} 0 \\ 1 \end{bmatrix}}^{B_m} u(k)$$

$$y(k) = \overbrace{\begin{bmatrix} 0,006564 & 0,002823 \end{bmatrix}}^{C_m} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix}$$

Berdasarkan Persamaan 2.7, kita dapat mengubah bentuk *state-space* diatas menjadi bentuk *augmented model.*

$$\overbrace{\begin{bmatrix} \Delta x_m(k+1) \\ y(k+1) \end{bmatrix}}^{x(k+1)} = \overbrace{\begin{bmatrix} A_m & 0_m^T \\ C_m A_m & 1 \end{bmatrix}}^{A} \overbrace{\begin{bmatrix} \Delta x_m(k) \\ y(k) \end{bmatrix}}^{x(k)} + \overbrace{\begin{bmatrix} B_m \\ C_m B_m \end{bmatrix}}^{B} \Delta u(k)$$

$$y(k) = \overbrace{\begin{bmatrix} 0_m & 1 \end{bmatrix}}^{C} \begin{bmatrix} \Delta x_m(k+1) \\ y(k+1) \end{bmatrix}$$

$$\overbrace{\begin{bmatrix} \Delta x_m(k+1) \\ y(k+1) \end{bmatrix}}^{x(k+1)} = \overbrace{\begin{bmatrix} 0 & 1 & 0 \\ 0,005205 & 0,9936 & 0 \\ 0 & 0,0094 & 1 \end{bmatrix}}^{A} \overbrace{\begin{bmatrix} \Delta x_m(k) \\ y(k) \end{bmatrix}}^{x(k)} + \overbrace{\begin{bmatrix} 0 \\ 1 \\ 0,002823 \end{bmatrix}}^{B} \Delta u(k)$$

$$y(k) = \overbrace{\begin{bmatrix} 0 & 0 & 1 \end{bmatrix}}^{C} \begin{bmatrix} \Delta x_m(k+1) \\ y(k+1) \end{bmatrix}$$

Matriks *A,B,C* yang merupakan bentuk *augmented model* akan digunakan selanjutnya dalam merancang sebuah kontroler MPC.

3.5.3 Penentuan Parameter dan *Gain* Kontroler MPC

Langkah selanjutnya dalam perancangan kontroler MPC adalah menentukan parameter dari kontroler MPC. Parameter yang dimaksud adalah *prediction horizon* (N_p), *control horizon* (N_c) dan *tuning parameter* pada indeks performansi (r_w). Pada perancangan kali ini, kita akan menggunakan parameter *prediction horizon* sebesar 5 langkah, *control horizon* senilai 2 langkah dan *tuning* parameter indeks performansi sebesar 1. Berdasarkan Persamaan 2.10 dan 2.11, nilai *output* terprediksi dan variabel kontrol yang akan datang dapat dihitung dengan menggunakan persamaan berikut:

$$Y = Fx(k_i) + \Phi\Delta U$$

Matriks F dan Φ dapat diformulasikan sebagai berikut:

$$F = \begin{bmatrix} CA \\ CA^2 \\ CA^3 \\ \vdots \\ CA^{N_p} \end{bmatrix} ; \Phi = \begin{bmatrix} CB & 0 & \cdots & 0 \\ CAB & CB & \cdots & 0 \\ CA^2B & CAB & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ CA^{N_p-1}B & CA^{N_p-2}B & \cdots & CA^{N_p-N_c}B \end{bmatrix}$$

Setelah nilai parameter kontroler ditentukan, maka kita dapat menentukan nilai Matriks F dan Φ sebagai berikut:

$$F = \begin{bmatrix} CA \\ CA^2 \\ CA^3 \\ CA^4 \\ CA^5 \end{bmatrix} ; \Phi = \begin{bmatrix} CB & 0 \\ CAB & CB \\ CA^2B & CA^1B \\ CA^3B & CA^2B \\ CA^4B & CA^3B \end{bmatrix} \quad (3.4)$$

Langkah selanjutnya adalah memasukkan nilai matriks *augmented model* ke dalam Persamaan 3.3. Hasil yang didapatkan dapat dilihat pada persamaan dibawah ini.

$$F = \begin{bmatrix} 0 & 0,0094 & 1 \\ 0,0001 & 0,0187 & 1 \\ 0,0001 & 0,0280 & 1 \\ 0,0002 & 0,0373 & 1 \\ 0,0002 & 0,0466 & 1 \end{bmatrix}; \Phi = \begin{bmatrix} 0,0028 & 0 \\ 0,0122 & 0,0028 \\ 0,0215 & 0,0122 \\ 0,0308 & 0,0215 \\ 0,0401 & 0,0308 \end{bmatrix}$$

Setelah mendapatkan matriks F dan Φ dapat, parameter selanjutnya yang akan dicari adalah *gain* dari kontroler MPC. *Gain* tersebut adalah K_{MPC} dan K_y. Untuk mencari nilai K_{MPC}, terlebih dahulu kita harus mencari nilai dari matriks $(\Phi^T\Phi + \overline{R})^{-1}(\Phi^T F)$.

$$Y = (\Phi^T\Phi + \overline{R})^{-1}(\Phi^T F) \quad (3.5)$$

$$Y = \begin{bmatrix} 0 & 0,0039 & 0,107 \\ 0 & 0,0026 & 0,0607 \end{bmatrix}$$

Nilai *gain* K_{MPC} merupakan baris pertama dari matriks *Y*. Oleh karena itu, nilai *gain* K_{MPC} dapat kita simpulkan sebagai matriks berikut ini:

$$K_{MPC} = \begin{bmatrix} 0 & 0,0039 & 0,107 \end{bmatrix}$$

Setelah *gain* K_{MPC} ditemukan, langkah selanjutnya adalah mencari *gain* K_y. Penguatan atau *gain* ini dapat ditemukan dari nilai matriks $(\Phi^T \Phi + \overline{R})^{-1}(\Phi^T \overline{R}_s)$.

$$Z = (\Phi^T \Phi + \overline{R})^{-1}(\Phi^T \overline{R_s}) \qquad (3.6)$$

$$Z = \begin{bmatrix} 0,107 \\ 0,607 \end{bmatrix}$$

Gain K_y merupakan baris pertama dari matriks *Z*. Berdasarkan matriks diatas, dapat kita ambil bahwa nilai K_y mempunyai nilai:

$$K_y = 0,107$$

3.5.4 Desain *State Estimation* Menggunakan *Observer*

Penggunaan *observer* pada sistem ini merupakan suatu langkah yang krusial. Ini dikarenakan adanya *state* yang tidak terukur pada sistem. Untuk mengatasinya, diperlukan sebuah *state estimation* berupa *observer* untuk mengestimasi nilai dari *state* yang tidak terukur ini.

Sebelum merancang sebuah *observer*, harus kita perhatian dahulu apakah sistem termasuk sistem *observability*. Artinya, semua *state* pada waktu $x(t_o)$ dapat ditentukan nilainya dari suatu *output* pada interval waktu tertentu [18]. Untuk menguji apakah suatu sistem *observable* atau tidak, sistem perlu diuji menggunakan matriks pada Persamaan 3.7 berikut.

$$rank\begin{bmatrix} C^T & A^T C^T & (A^T)^2 C^T \end{bmatrix} = n \qquad (3.7)$$

Variabel *n* merupakan jumlah *state* yang ada pada sistem. Pada Persamaan 3.5, matriks sistem yang digunakan adalah matriks yang

berdasarkan pada *augmented model*. Berdasarkan perhitungan pada MATLAB, nilai *rank* matriks diatas bernilai 3, sama dengan jumlah *state* pada *augmented model*. Dapat disimpulkan bahwa sistem *observable* dan dapat ditentukan setiap nilai *state*-nya pada waktu $x(t_o)$.

Setelah menentukan bahwa sistem *observable*, langkah selanjutnya adalah mendesain *observer*. Perlu diketahui, bahwa pada perancangan *observer*, matriks yang digunakan adalah matriks yang berasal dari *augmented model*. Jenis *observer* yang digunakan adalah *full state order observer*. Oleh karena itu, perlu ditemukan nilai K_e yang merupakan *gain observer*. *Gain observer* ini dapat ditemukan melalui beberapa metode, salah satunya metode *Ackermann* seperti yang digunakan pada Tugas Akhir ini. Pencarian nilai *gain observer* K_e menggunakan metode *Ackermann* dapat dilihat pada Persamaan 3.8 dan Persamaan 3.9.

$$K_e = \phi(A)\begin{bmatrix} C \\ CA \\ CA^2 \end{bmatrix}^{-1}\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{3.8}$$

$$\phi(A) = (s-\mu_1)(s-\mu_2)(s-\mu_3) \tag{3.9}$$

Variabel μ merupakan nilai *eigenvalue* yang diinginkan pada sistem. Dengan metode *trial-and-error*, *pole* yang diinginkan berada pada -0,001; -0,12 dan -0,03. Sehingga dengan bantuan software MATLAB, kita dapat menentukan nilai *gain observer* sebesar:

$$K_e = \begin{bmatrix} 121{,}9463 \\ 122{,}1254 \\ 2{,}1446 \end{bmatrix}$$

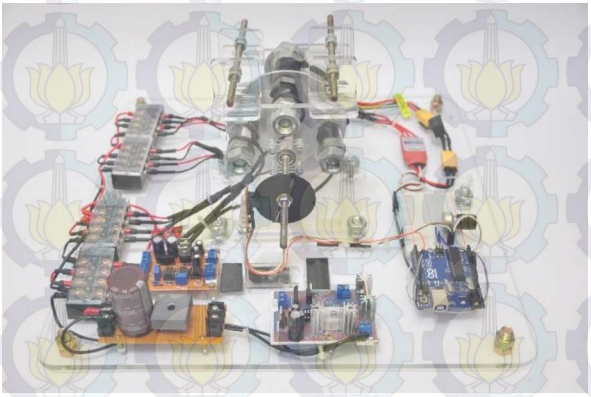
BAB 4
PENGUJIAN DAN ANALISA

Bab 4 menjelaskan tentang pengujian sistem berupa simulasi menggunakan software MATLAB dengan menggunakan pemodelan yang sudah didapatkan. Setelah mengetahui hasil simulasi, kontroler MPC akan diimplementasikan ke dalam sistem dan hasilnya akan dianalisa dengan menggunakan parameter karakteristik respon.

4.1 Gambaran Umum Pengujian Sistem

Pada pengerjaan Tugas Akhir ini, akan dilakukan beberapa pengujian pada sistem dengan beberapa kondisi pembebanan. Perancangan sistem seperti telah dijelaskan pada Bab 3 dapat direalisasikan menjadi sebuah sistem *plant* pengaturan kecepatan motor BLDC seperti terlihat pada Gambar 4.1. Sebelumnya, akan terlebih dahulu dilakukan pengujian terhadap beberapa komponen sistem. Setelah itu, akan dilakukan pengujian *open loop* pada motor BLDC dengan rentang kecepatan 1500-2500 RPM.

Tahapan selanjutnya adalah melakukan simulasi dari kontroler MPC. Simulasi akan dilakukan pada *software* MATLAB dengan bantuan *software* Simulink. Fokus utama dalam simulasi sistem ini adalah mengaplikasikan kontroler MPC pada sistem dengan objektif kontrol berupa *tracking* dengan nilai *setpoint* yang berubah-ubah.



Gambar 4.1 Realisasi *Plant* atau Sistem Pengaturan Kecepatan Motor *Brushless DC.*

Langkah terakhir dalam pengujian sistem adalah implementasi kontroler MPC pada sistem sesungguhnya. Akan dilakukan beberapa analisa dengan membandingkan respon yang didapat antara simulasi dan implementasi. Selain itu, akan dilakukan juga analisa pengaruh antara sebelum dan sesudah pemberian kontroler MPC pada sistem.

4.2 Pengujian Perangkat Keras

Pengujian perangkat keras pada pelaksanan Tugas Akhir kali ini bertujuan untuk mengetahui apakah perangkat keras yang berjalan pada sistem dapat bekerja secara semestinya. Pengujian perangkat keras tersebut meliputi pengujian *driver* motor BLDC berupa *Electronic Speed Control* (ESC).

4.2.1 Pengujian *Electronic Speed Control* Motor BLDC

Driver motor BLDC atau yang biasa disebut *Electronic Speed Control* (ESC) merupakan salah satu perangkat keras yang mempuyai kinerja paling krusial dalam menggerakkan motor BLDC. Oleh karena itu, penting untuk mengetahui dan memastikan ESC memiliki performa yang baik untuk menjalakan motor BLDC dengan baik.

Seperti yang telah dijelakan pada Bab 3, *Input* yang diberikan sebagai masukan ke dalam ESC ini mempunyai frekuensi sebesar 50 Hz atau 2 ms (*milisecond*). Untuk kecepatan minimal, nilai sinyal PWM yang masuk ke dalam ESC bernilai 1 ms. Sedangkan untuk kecepatan maskimal, dibutuhkan sinyal PWM sebesar 2 ms. Rentang nilai 1-2 ms tersebut di *map* atau diberi rentang baru bernilai 0-1000. Ini bertujuan untuk memudahkan pemberian *input* nilai atau masukan sinyal kontrol pada saat simulasi maupun implementasi. Nilai tegangan dari *input* PWM ini juga dihitung untuk mengetahui karakteristik dari motor BLDC. Hubungan *input* PWM, tegangan *output* driver dalam satuan milivolt dan kecepatan motor BLDC dapat dilihat pada Tabel 4.1

Berdasarkan Tabel 4.1 dapat kita dapat melihat bahwa hubungan tegangan *output* ESC linear dengan *input* PWM yang diberikan. Oleh karena itu, dapat disimpulkan bahwa *driver* ESC dapat digunakan untuk menggerakkan motor BLDC.

Tabel 4.1 Hubungan Nilai *Input* PWM, Tegangan *Output* ESC dan Kecepatan Motor BLDC.

Input PWM	Tegangan (mV)	Kecepatan Motor BLDC (RPM)
50	277,6	340
100	286,2	1099
150	294,2	1722
200	302,5	2196
250	310,8	2503
300	319,2	2751
350	327,3	2905
400	335,7	3043
450	343,9	3115
500	352,2	3199
550	360,2	3236
600	368,6	3300
650	376,7	3319
700	385,2	3363
750	393,3	3370
800	402,1	3409
850	410,2	3411
900	418,5	3471
950	426,8	3512
1000	435,4	3529

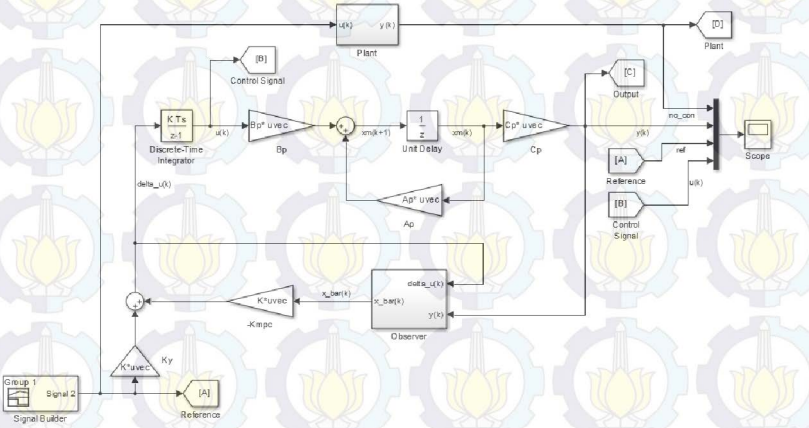
4.3 Simulasi Sistem

Simulasi merupakan salah satu langkah penting sebelum mengimplementasikan kontroler pada sistem yang sesungguhnya. Dengan hasil simulasi sesuai dengan performa yang diharapkan, pengimplementasian kontroler pada sistem akan menjadi lebih mudah dan mencapai kriteria atau performa yang diinginkan.

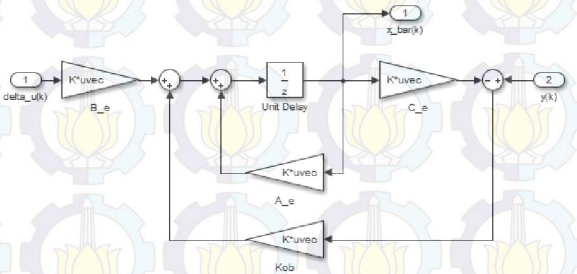
4.3.1 Diagram Blok Simulasi Sistem

Pengerjaan simulasi pada pelaksanaan Tugas Akhir kali ini akan dibantu oleh program MATLAB dan Simulink. Secara garis besar, diagram blok simulasi sistem terdiri dari *state-space* sistem, penguatan atau *gain* MPC dan subsistem *observer*. Blok diagram dari simulasi sistem dapat dilihat pada Gambar 4.2.

Seperti terlihat pada Gambar 4.2, terdapat blok diagram untuk *state-space* sistem yang terdiri dari blok *gain* A_p, B_p dan C_p. Selain itu terdapat blok penguatan pada kontroler MPC, yaitu K_y dan K_{MPC}. beserta sebuah blok *integrator* diskrit. Selain itu, ditambahkan blok *signal builder* sebagai penghasil nilai referensi atau *setpoint* pada sistem.Selain sistem diatas, ditambahkan pula satu subsistem berupa *observer*. Subsistem *observer* digunakan sebagai *state estimator* untuk mengestimasi suatu nilai *state* yang tidak diketahui. Subsistem *observer* dapat dilihat pada Gambar 4.3.



Gambar 4.2 Diagram Blok Simulasi Pengujian Kontroler MPC pada Sistem.



Gambar 4.3 Subsistem *Observer* pada Simulasi Sistem

4.3.2 Prosedur Pengerjaan Simulasi Sistem

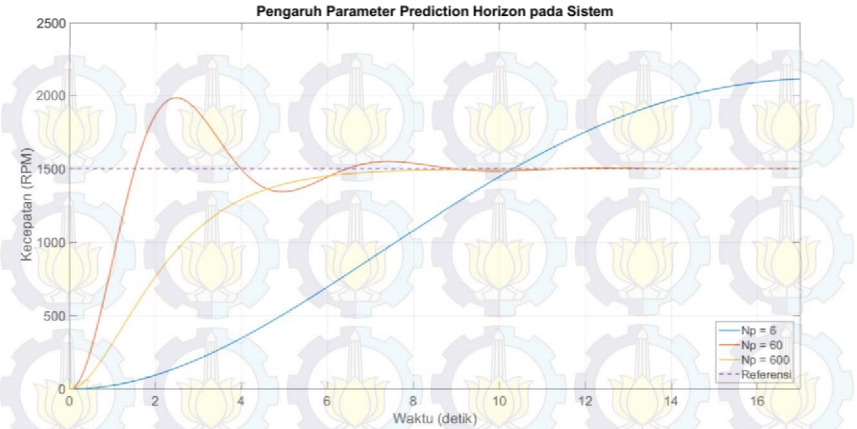
Pada langkah pengerjaan simulasi sistem, ada beberapa nilai dan variabel yang diamati pengaruhnya pada sistem. Variabel tersebut adalah nilai *prediction horizon* (N_p), *control horizon* (N_c) dan nilai *tuning parameter* pada indeks performansi (r_w). Ketiga variabel tersebut akan diuji satu persatu mengenai seberapa besar pengaruh ketiga variabel tersebut pada sistem. Untuk pengujian, sistem akan diberikan 2 input yang berbeda, yaitu *input unit step* dan *input tracking* sistem. Pada *tracking* sistem, nilai *setpoint* akan berubah-ubah dengan mengikuti *input ramp* pada kondisi kecepatan berbeda dan mengamati performa kontroler MPC dalam mengikuti dari nilai *setpoint* tersebut.

Setelah respon dari sistem yang diberi kontroler terlihat, maka langkah selanjutnya adalah menentukan karakteristik respon dari sistem tersebut. Parameter pada *input step* yang akan diamati adalah nilai *steady-state error*, *overshoot*, *settling time* dan *rise time*. Nilai *rise time* yang digunakan saat respon menyentuh angka 5% hingga 95%. Sedangkan *settling time* yang akan dihitung berada pada saat 2% dari nilai *steady state*. Sedangkan pada *input tracking* akan dilihat performa kontroler berdasarkan nilai *error steady-state* sistem.

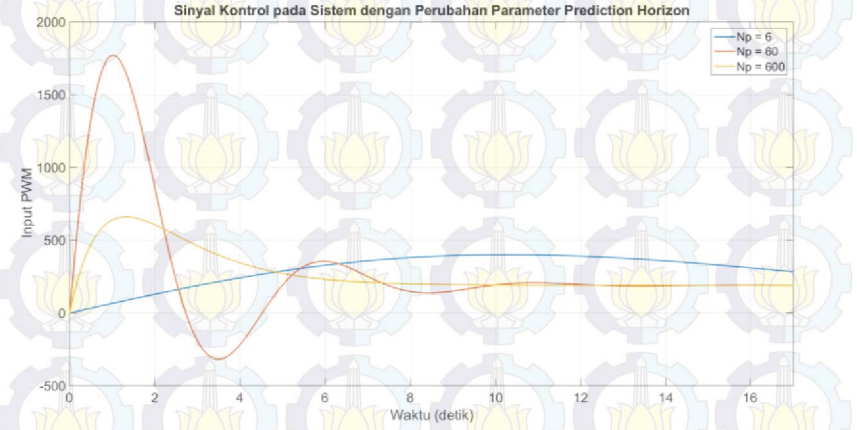
4.4 Pengaruh Parameter N_p pada Sistem

Variabel pertama yang akan diuji pada sistem dan kontroler MPC adalah nilai *prediction horizon* atau N_p. Nilai *prediction horizon* yang dijadikan parameter mempunyai nilai yang bervariasi, mulai dari satuan hingga ratusan. Setelah itu, akan dilihat karakteristik dari respon simulasi untuk menentukan nilai *prediction horizon* yang akan digunakan pada saat implementasi.

Terlihat dari Gambar 4.4, bahwa respon yang dihasilkan oleh sistem akan berbeda-beda seiring dengan parameter *prediction horizon* yang berbeda pula. Sedangkan sinyal kontrol yang dihasilkan oleh kontroler MPC dapat dilihat pada Gambar 4.5. Pada pengujian *prediction horizon* ini, nilai *control horizon* dan *tuning parameter* akan dijadikan nilai tetap, untuk melihat pengaruh dari perubahan nilai *prediction horizon* pada sistem. Pada pembebanan minimal dan nilai N_p = 6, respon yang dihasilkan sama sekali tidak mengikuti respon *unit step*. Respon terlihat baru mengikuti *tracking setpoint* yang diberikan ketika nilai *prediction horizon* bernilai dua digit ke atas, dalam hal ini diambil nilai 60 dan 600.



Gambar 4.4 Pengaruh Parameter *Prediction Horizon* pada Sistem.



Gambar 4.5 Sinyal Kontrol pada Sistem dengan Perubahan Parameter *Prediction Horizon*.

Pada nilai N_p = 60, terlihat bahwa respon sudah mengikuti *unit step* yang diberikan. Tetapi, seperti terlihat pada Gambar 4.4, *overshoot* yang dihasilkan oleh sistem sangat tinggi. Tentu saja respon seperti ini sangat dihindari karena akan merusak motor. Oleh karena itu, perlu di-*tuning* lagi

nilai dari *prediction horizon* agar respon yang dih.asilkan tidak mengalami *overshoot*.

Setelah melakukan beberapa kali pengujian, akhirnya ditemukan nilai $N_p = 600$. Respon yang dihasilkan jauh lebih baik dibandingkan nilai $N_p = 60$. Respon yang dihasilkan tidak mengalami *overshoot* sama sekali. Akan tetapi, terlihat bahwa respon yang dihasilkan lebih lambat daripada nilai $N_p = 60$. Untuk melihat karakteristik respon sistem dari tiap pembebanan, dapat dilihat pada subbab dibawah ini.

4.4.1 Beban Minimal

Pada pembebanan minimal, nilai *prediction horizon* yang diuji berada pada nilai 5, 50 dan 500. Untuk 2 parameter lainnya, seperti *control horizon* dan *tuning parameter* akan diberi nilai tetap dengan $N_c = 2$ dan $r_w = 10$. Tabel 4.2 menyajikan data mengenai nilai karakteristik respon untuk tiap-tiap nilai parameter *predicion horizon*.

Tabel 4.2 Karakteristik Respon Sistem pada Pembebanan Minimal untuk Tiap-Tiap Variabel *Prediction Horizon*.

N_p	*Steady-State Error (%)*	*Rise Time* (Detik)	*Settling Time* (Detik)	Presentase *Overshoot (%)*
5	Tidak mengikuti referensi			
50	0	1,25	6,76	33,26
500	0	7,66	10,34	0

Terlihat pada Tabel 4.2, respon sistem dengan nilai $N_p = 5$ tidak mengikuti referensi *unit step*. Sedangkan nilai *prediction horizon* sebesar 50 memiliki respon yang lebih cepat dibandingkan dengan sistem yang mempunyai nilai *prediction horizon* 500. Akan tetapi, respon yang cepat ini memiliki kelemahan yang sangat menonjol, yaitu nilai *overshoot*-nya yang tinggi.

4.4.2 Beban Nominal

Prosedur pengujian respon sistem pada beban nominal mempunyai langkah yang sama seperti pada pembebanan minimal. Hanya saja, nilai *prediction horizon* yang diuji pada sistem mempunyai nilai 4, 40 dan 400. Sedangkan nilai *control horizon* dan *tuning parameter* dibuat tetap pada nilai $N_c = 3$ dan $r_w = 20$. Karakteristik respon pada pembebanan nominal dapat dilihat pada Tabel 4.3.

Tabel 4.3 Karakteristik Respon Sistem pada Pembebanan Nominal untuk Tiap-Tiap Variabel *Prediction Horizon*.

N_p	*Steady-State Error (%)*	*Rise Time* (Detik)	*Settling Time* (Detik)	Presentase *Overshoot (%)*
4	Tidak mengikuti referensi			
40	0	1,62	16,77	52,57
400	0	5,70	7,74	0

Respon sistem pada pembebanan nominal mempunyai kemiripan respon yang sama seperti pembebanan minimal jika nilai *prediction horizon*-nya diubah-ubah. Terlihat pada Tabel 4.3, respon sistem dengan nilai $N_p = 4$ tidak mengikuti referensi *unit step*. Sedangkan nilai *prediction horizon* sebesar 40 memiliki respon yang lebih baik dibandingkan *prediciton horizon* 4 walaupun terdapat nilai *overshoot* yang lebih tinggi. Untuk mengkompensasi nilai *overshoot* yang tinggi ini, maka nilai N_p dinaikkan hingga mencapai 400. Respon yang dihasilkan tidak mempunyai *overshoot* sama sekali, walaupun nilai *rise time* sistem menjadi lebih besar. Artinya, respon akan menjadi lebih lambat.

4.4.3 Beban Maksimal

Pada pengujian nilai *prediction horizon* terakhir, akan dilakukan pada pembebanan maksimal. Nilai *prediction horizon* yang digunakan pada sistem mempunyai nilai 6, 60 dan 600. Sedangkan nilai *control horizon* dan *tuning parameter* dibuat tetap pada nilai $N_c = 4$ dan $r_w = 30$. Karakteristik respon pada pembebanan nominal dapat dilihat pada Tabel 4.4 berikut ini.

Pada, Tabel 4.4, dapat diamati bahwa respon yang dihasilkan pada tiap-tiap nilai *prediction horizon* mempunyai karakteristik yang serupa seperti pada pembebanan minimal dan nominal. Pada variabel *prediction*

Tabel 4.4 Karakteristik Respon Sistem pada Pembebanan Maksimal untuk Tiap-Tiap Variabel *Prediction Horizon*.

N_p	*Steady-State Error (%)*	*Rise Time* (Detik)	*Settling Time* (Detik)	Presentase *Overshoot (%)*
6	Tidak mengikuti referensi			
60	0	1,52	8,18	32,28
600	0	5,01	6,65	0

horizon yang paling rendah, respon masih tidak mengikuti referensi. Pada *prediction horizon* 30 dan 300 barulah respon sistem mengikuti referensi dari *unit step*. Respon dengan nilai *prediction horizon* 30 mempunyai respon yang lebih cepat walaupun nilai *overshoot* yang tinggi. Sedangkan nilai *prediction horizon* 300 mempunyai respon yang lebih lambat, tetapi tidak memiliki nilai *overshoot*.

4.5 Pengaruh Parameter N_c pada Sistem

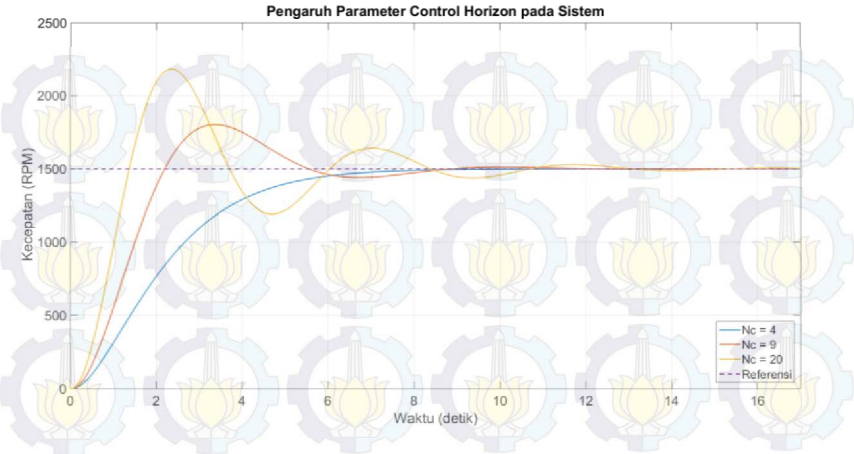
Setelah menentukan pengaruh dari *prediction horizon* pada srespon sistem, variabel selanjutnya yang akan diuji adalah pengaruh dari *control horizon* atau N_c pada respon sistem. Sama seperti *prediction horizon*, nilai dari *control horizon* yang diuji bervariasi dari satuan hingga ratusan. Nilai *control horizon* yang paling baik respon sistemnya akan dimasukkan sebagai parameter pada implementasi. Gambar 4.6 menunjukkan pengaruh dari perubahan nilai *control horizon* pada respon sistem.

Pada pengujian control *horizon* ini, nilai *prediction horizon* dan *tuning parameter* akan dijadikan nilai tetap, untuk melihat pengaruh dari perubahan nilai *control horizon* pada sistem. Terlihat pada gambar bahwa semua nilai variabel *control horizon* mengikuti dari *unit step*. Hanya saja, terlihat bahwa semakin kecil nilai dari *control horizon*, maka respon akan semakin lambat dan respon yang dihasilkan mirip seperti orde 1. Sebaliknya, semakin besar nilai *control horizon*, maka respon dari sistem akan semakin cepat. Akan tetapi, nilai *control horizon* yang semakin tinggi akan mengakibatkan semakin tinggi pula *overshoot* yang dihasilkan. Sinyal kontrol yang dihasilkan akibat perubahan dari parameter *control horizon* dapat dilihat pada Gambar 4.7. Untuk melihat pengaruh dari *control horizon* pada setiap pembebanan, dapat dilihat pada subbab di bawah ini.

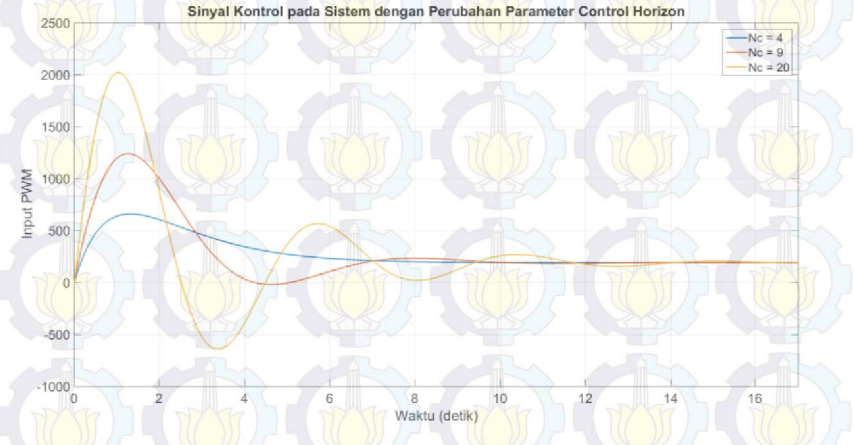
4.5.1 Beban Minimal

Pada pembebanan minimal, nilai *control horizon* yang diuji berada pada nilai 2, 10 dan 25. Untuk 2 parameter lainnya, seperti *prediction horizon* dan *tuning parameter* akan diberi nilai tetap dengan $N_p = 500$ dan $r_w = 10$. Tabel 4.5 menyajikan data mengenai nilai karakteristik respon untuk tiap-tiap nilai parameter *control horizon*.

Terlihat pada Tabel 4.5, semakin kecil nilai dari *control horizon*, semakin baik pula respon yang dihasilkan dilihat dari nilai *overshoot*.



Gambar 4.6 Pengaruh *Control Horizon* pada Sistem.



Gambar 4.7 Sinyal Kontrol pada Sistem dengan Perubahan Parameter *Control Horizon*.

Akan tetapi, akibat dari semakin kecilnya nilai *overshoot*, waktu respon yang dihasilkan menjadi semakin lambat. Akan tetapi, dipilih nilai *control horizon* yang paling kecil dikarenakan respon dengan *control horizon* paling kecil tidak menghasilkan *overshoot* sama sekali. Tentu

saja respon ini dibutuhkan karena respon dengan *overshoot* yang tinggi tentu dihasilkan dari sinyal kontrol yang tinggi juga, yang dapat berujung pada rusaknya motor BLDC.

Tabel 4.5 Karakteristik Respon Sistem pada Pembebanan Minimal untuk Tiap-Tiap Variabel *Control Horizon*.

N_c	*Steady-State Error (%)*	*Rise Time (Detik)*	*Settling Time* (Detik)	Presentase *Overshoot (%)*
2	0	7,66	10,37	0
10	0	1,53	8,32	33,26
25	0	1,12	8,07	49,80

4.5.2 Beban Nominal

Setelah mengetahui respon dari pembebanan minimal, langkah selanjutnya adalah menguji pengaruh *control horizon* pada pembebanan nominal. Nilai *control horizon* yang akan dimasukkan berada pada nilai 5, 15 dan 30. Sedangkan nilai *prediction horizon* dan *tuning parameter* akan dibuat konstan dengan N_p = 400 dan r_w = 20. Karakteristik respon pada sistem dengan pembebanan nominal dapat dilihat pada Tabel 4.6 berikut ini.

Terlihat pada Tabel 4.6, semakin kecil nilai dari *control horizon*, semakin baik pula respon yang dihasilkan dilihat dari nilai *overshoot*. Ini terlihat pada nilai *control horizon* paling rendah, respon yang diberikan menyerupai orde 1 dan tidak terdapat *overshoot*. Akan tetapi, nilai *rise time* pada *control horizon* bernilai 3 mempunyai nilai yang lebih besar dan berakibat pada respon yang lambat. Respon lambat ini bisa dibuat semakin cepat dengan meningkatkan nilai *control horizon*. Akan tetapi, peningkatan respon dari sistem bakat berakibat pada semakin tinggi pula nilai dari *overshoot* sistem.

Tabel 4.6 Karakteristik Respon Sistem pada Pembebanan Nominal untuk Tiap-Tiap Variabel *Control Horizon*.

N_c	*Steady-State Error (%)*	*Rise Time (Detik)*	*Settling Time* (Detik)	Presentase *Overshoot (%)*
3	0	5,70	7,74	0
15	0	1,74	11,64	37,13
30	0	1,43	12,87	46,67

Untuk implementasi pada pembebanan nominal, akan digunakan parameter *control horizon* sebesar 3. Ini dikarenakan respon tidak mengalami *overshoot* yang akan mengakibatkan gangguan pada sistem.

4.5.3 Beban Maksimal

Pengujian *control horizon* terakhir adalah pengujian pada sistem dengan pembebanan maksimal. Nilai *control horizon* yang digunakan berada pada nilai 4, 9 dan 20. Nilai *prediction horizon* dan *tuning parameter* akan dibuat konstan dengan $N_p = 600$ dan $r_w = 30$. Karakteristik respon pada sistem dengan pembebanan nominal dapat dilihat pada Tabel 4.7 berikut ini.

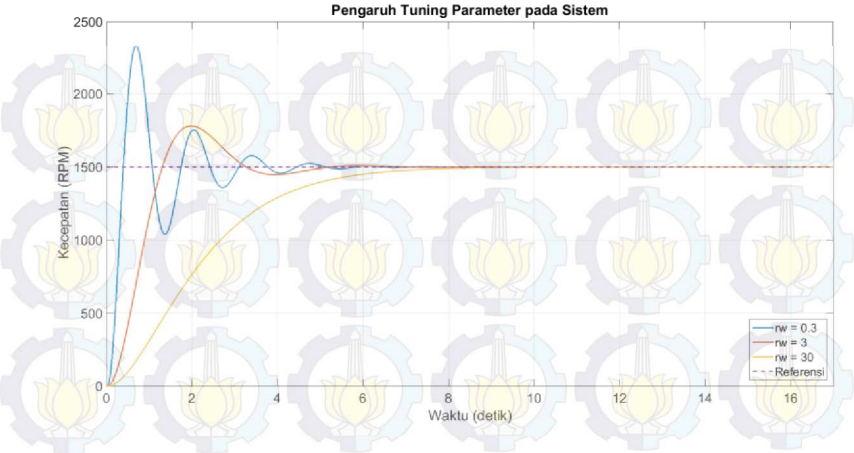
Tabel 4.7 Karakteristik Respon Sistem pada Pembebanan Maksimal untuk Tiap-Tiap Variabel *Control Horizon*.

N_c	*Steady-State Error (%)*	*Rise Time (Detik)*	*Settling Time* (Detik)	Presentase *Overshoot (%)*
4	0	5,01	6,65	0
9	0	2,19	7,93	20,15
20	0	1,35	11,69	45,47

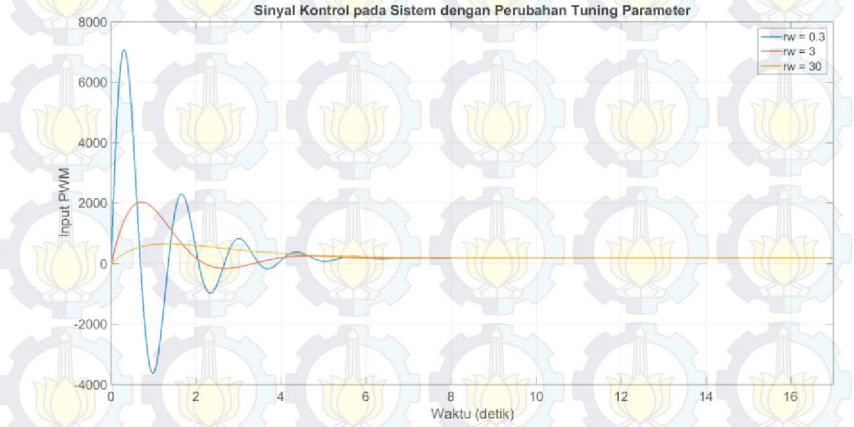
Tabel 4.7 menunjukkan, bahwa respon sistem yang paling baik dihasilkan oleh sistem dengan nilai *control horizon* sebesar 4. Respon dikatakan baik karena respon sistem yang dihasilkan tidak memiliki *overshoot* walaupun respon sistem lebih lambat. Cara untuk menaikkan waktu respon sistem dapat ditempuh dengan menaikkan nilai *control horizon*. Akan tetapi, perlu berhati-hati dalam meningkatkan waktu respon dari sistem. Semakin cepat respon, maka semakin tinggi pula *overshoot* yang dihasilkan. Ini terlihat dari nilai *control horizon* 9 dan 20 yang menunjukkan semakin rendah *rise time* yang dihasilkan, semakin tinggi pula *overshoot* yang dihasilkan.

4.6 Pengaruh Parameter r_w pada Sistem

Variabel terakhir yang diuji pengaruhnya pada sistem adalah nilai *tuning parameter* r_w pada indeks performansi J. Nilai dari *tuning parameter* ini mempunyai kemiripan dengan matriks pembobot Q dan R. pada kontroler LQR. Sama seperti prosedur pengujian sebelumnya, nilai dari *prediction horizon* dan *control horizon* akan dibuat konstan untuk mengetahui pengaruh dari *tuning parameter* pada indeks performansi dan sinyal kontrolnya, seperti terlihat pada Gambar 4.8 dan Gambar 4.9.



Gambar 4.8 Pengaruh Parameter *Tuning* Indeks Performansi pada Sistem



Gambar 4.9 Sinyal Kontrol pada Sistem dengan Perubahan *Tuning* Parameter pada Indeks Performansi.

Gambar 4.8 menunjukkan pengaruh dari perubahan nilai *tuning parameter* pada respon sistem. Terlihat pada gambar bahwa semua nilai variabel *tuning parameter* mengikuti dari *unit step*. Terlihat bahwa

semakin kecil nilai r_w, semakin cepat respon yang didapatkan. Sebaliknya, semakin besar nilai r_w, maka respon dari sistem akan semakin lambat. Akan tetapi, nilai *tuning parameter* yang semakin kecil akan mengakibatkan meningkatnya *overshoot* yang dihasilkan.

4.6.1 Beban Minimal

Pengujian *tuning parameter* pertama ada pada kondisi pembebanan minimal. Nilai *tuning parameter* yang digunakan berada pada nilai 0,1; 1 dan 10. Nilai *prediction horizon* dan *control horizon* akan dibuat konstan dengan N_p = 500 dan N_c = 2. Karakteristik respon pada sistem dengan pembebanan nominal dapat dilihat pada Tabel 4.8 berikut ini.

Tabel 4.8 Karakteristik Respon Sistem pada Pembebanan Minimal untuk Tiap-Tiap Variabel *Tuning Parameter*.

r_w	*Steady-State Error (%)*	*Rise Time* (Detik)	*Settling Time* (Detik)	Presentase *Overshoot (%)*
0.1	0	0,58	3,07	30
1	0	2,69	3,52	0
10	0	7,63	10,4	0

Tabel 4.8 menunjukkan bahwa semakin kecil nilai *tuning parameter*, semakin cepat pula respon dari sistem. Ini diperlihatkan dari nilai *rise time* yang semakin kecil. Akan tetapi, nilai *tuning parameter* yang semakin kecil akan mengakibatkan munculnya *overshoot* pada sistem. Untuk mengurangi dan meminimalkan *overshoot* sistem, maka nilai dari *tuning parameter* ini dinaikkan sedemikian sehingga *overshoot* semakin berkurang. Terlihat pada nilai r_w yang tinggi, pada nilai 1 dan 10, tidak terdapat *overshoot* pada sistem. Akan tetapi, apabila dibandingkan nilai *rise time* pada nilai r_w sama dengan 1 dan 10, respon akan semakin cepat apabila nilai *tuning parameter* semakin kecil. Oleh karena itu, dipilih nilai r_w sebesar 1 dengan pertimbangan tidak adanya respon *overshoot* dan *rise time* yang cepat.

4.6.2 Beban Nominal

Pengujian selanjutnya adalah melihat pengaruh *tuning parameter* pada pembebanan nominal. Rentang nilai *tuning parameter* yang digunakan pada pengujian berada pada nilai 0,2; 2, dan 20. Karakteristik respon dapat dilihat pada Tabel 4.9.

Tabel 4.9 Karakteristik Respon Sistem pada Pembebanan Nominal untuk Tiap-Tiap Variabel *Tuning Parameter*.

r_w	*Steady-State Error (%)*	*Rise Time* (Detik)	*Settling Time* (Detik)	Presentase *Overshoot (%)*
0,2	0	0,56	3,85	38,7
2	0	2,48	4,11	3,40
20	0	5,70	7,74	0

Terlihat pada Tabel 4.9 bahwa perubahan nilai *tuning parameter* mempunyai pengaruh yang cukup signifikan. Semakin kecil nilai *tuning parameter*, semakin cepat respon yang dihasilkan sistem walaupun nilai *overshoot* yang semakin tinggi. Untuk mengkompensasi nilai *overshoot* yang tinggi, maka nilai *tuning parameter* perlahan dinaikkan walaupun respon sistem semakin lambat.

4.6.3 Beban Maksimal

Pengujian *tuning parameter* terakhir adalah pengujian pada sistem dengan pembebanan maksimal. Nilai *tuning parameter* yang digunakan berada pada nilai 0,3; 3 dan 30. Sedangkan nilai *prediction horizon* dan *control horizon* akan dibuat konstan dengan N_p = 600 dan N_c = 4. Karakteristik respon pada sistem dengan pembebanan maksimal dapat dilihat pada Tabel 4.10 berikut ini.

Tabel 4.10 menunjukkan bahwa nilai *tuning parameter* yang semakin besar akan meningkatkan waktu respon dari sistem. Ini terlihat dari nilai *rise time* yang semakin kecil pada nilai *tuning parameter* yang semakin kecil. Akan tetapi, penggunaan nilai r_w yang telalui kecil sebaiknya di hindari dikarenakan adanya *overshoot* yang dihasilkan sistem. Nilai *overshoot* ini mungkin terlihat semakin besar, ketika nilai r_w semakin kecil.

Tabel 4.10 Karakteristik Respon Sistem pada Pembebanan Maksimal untuk Tiap-Tiap Variabel *Tuning Parameter*.

r_w	*Steady-State Error (%)*	*Rise Time* (Detik)	*Settling Time* (Detik)	Presentase *Overshoot (%)*
0,3	0	0,39	4,25	55,35
3	0	1,30	4,59	18,77
30	0	5,01	6,65	0

Oleh karena itu, alternatif lain ada pada penggunaan nilai r_w = 30. Hal ini dikarenakan nilai *tuning parameter* mempunyai respon yang menyerupai orde 1 dan tidak adanya *overshoot*. Walaupun begitu, nilai *rise time* yang dihasilkan semakin lambat. Oleh karena itu, untuk pengujian implementasi digunakan nilai *tuning parameter* sebesar 30.

4.7 Respon *Tracking* Sistem

Setelah mengetahui respon sistem setelah diberikan masukan *unit step*, analisa selanjutnya adalah mengetahui respon sistem saat diberikan *input* berupa referensi *tracking*. Referensi *tracking* yang diberikan meliputi sinyal *uji ramp* menuju kecepatan tertentu. Untuk keperluan analisa, akan diuji berapa besar pengaruh *prediction horizon, control horizon* dan *tuning parameter* ketika sistem diberi referensi *tracking* pada setiap pembebanan. Untuk analisa dari respon *tracking* sistem, akan diuji seberapa besar nilai *error steady-state* pada *unit ramp*, nilai RMSE (*Root Mean Square Error*), dan *overshoot* dari respon *tracking* yang dihasilkan.

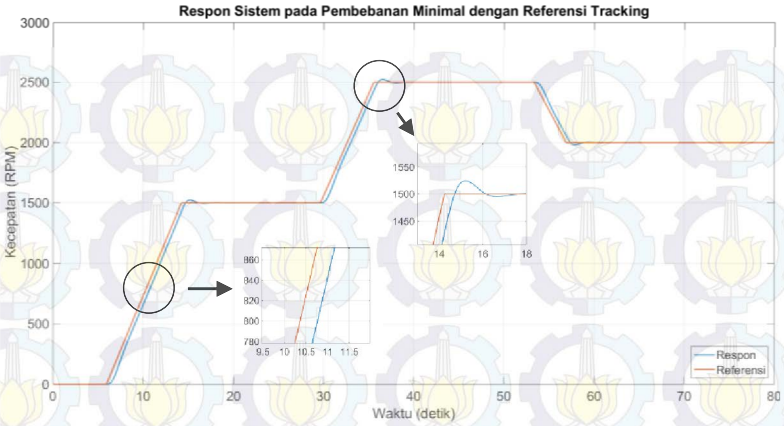
4.7.1 Beban Minimal

Pada pembebanan minimal, parameter kontroler MPC yang akan diberikan pada sistem bernilai N_p = 50; N_c = 2 dan r_w = 0,1. Setelah hasil didapatkan, barulah respon dianalisa karakteristik responnya seperti terlihat pada Gambar 4.10.

Terlihat pada Gambar 4.10, respon yang dihasilkan sudah dapat mengikuti referensi *tracking* yang diberikan. Hanya saja, terlihat bahwa terdapat nilai perbedaan antara respon dan referensi saat sistem diberikan *input unit ramp*. Selain itu, terdapat pula nilai *overshoot* ketika sistem telah memasuki nilai konstan pada kecepatan 1.500, 2.000 dan 2.500 RPM. Akan tetapi, nilai *error* tersebut sangatlah kecil sehingga dapat diabaikan. Untuk lebih jelasnya, karakteristik respon saat diberi referensi *tracking* dapat dilihat pada Tabel 4.11.

Tabel 4.11 Karakteristik Respon Sistem pada Pembebanan Minimal dengan Referensi *Tracking*.

Parameter	Nilai
Steady-State Error (%)	0,057
Presentase *Overshoot* (%)	0,017
RMSE	0,096



Gambar 4.10 Respon Sistem pada Pembebanan Minimal dengan Referensi *Tracking*.

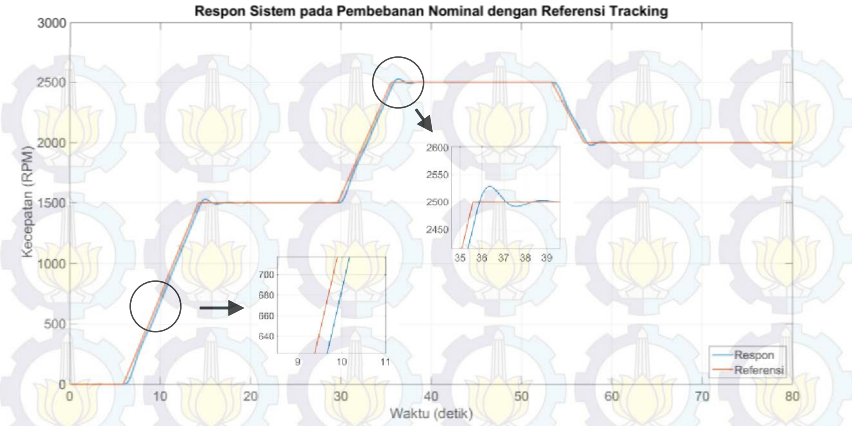
4.7.2 Beban Nominal

Setelah mendapatkan respon *tracking* pada pembebanan minimal, analisa selanjutnya adalah mengamati respon sistem pada pembebanan nominal jika sistem diberi referensi *tracking*. Pada pembebanan nominal, akan diberikan parameter kontroler MPC yang berbeda dengan pembebanan minimal. Parameter tersebut berupa N_p = 40; N_c = 3 dan r_w = 0,2. Hasil respon pembebanan nominal dapat dilihat pada Gambar 4.11.

Terlihat pada Gambar 4.11, respon yang dihasilkan sistem sudah mengikuti referensi *tracking* yang diberikan. Masalah yang sama seperti pada pembebanan minimal timbul pula pada respon di pembebanan nominal. Masih terdapat perbedaan nilai dan respon saat sistem mengikuti referensi *unit ramp* pada pada sistem. Selain itu, masih terdapat *overshoot* pada sistem ketika respon memasuki nilai referensi konstan. Karakteristik respon pada pembebanan no minal dapat dilihat pada Tabel 4.12.

Tabel 4.12 Karakteristik Respon Sistem pada Pembebanan Nominal dengan Referensi *Tracking*.

Parameter	Nilai
Steady-State Error (%)	0,041
Presentase *Overshoot* (%)	0,020
RMSE	0,092



Gambar 4.11 Respon Sistem pada Pembebanan Nominal dengan Referensi *Tracking*.

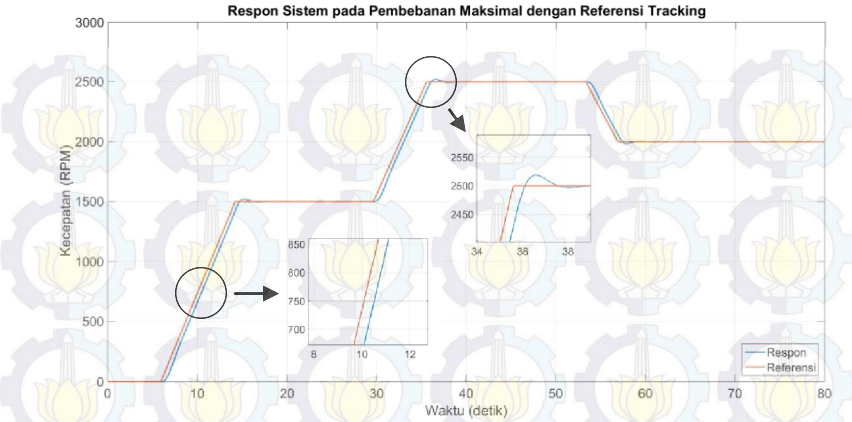
4.7.3 Beban Maksimal

Pengamatan terakhir untuk referensi tracking dilakukan pada pembebanan maksimal. Pada pembebanan maksimal, akan diberikan parameter kontroler MPC yang berbeda dengan pembebanan minimal maupun nominal. Parameter tersebut berupa $N_p = 60$; $N_c = 4$ dan $r_w = 0{,}3$. Hasil respon pada pembebanan maksimal dapat dilihat pada Gambar 4.12.

Respon yang terlihat pada Gambar 4.12 sudah dapat mengikuti referensi *tracking* yang diberikan. Selain itu, masih terdapat perbedaan respon ketika sistem mengikuti sinyal *unit ramp*. *Overshoot* pun masih terjadi ketika respon mulai memasuki nilai referensi konstan. Akan tetapi, nilai kedua parameter karakteristik respon tersebut sangatlah kecil hingga dapat dianggap tidak terlalu mempengaruhi performa dari motor BLDC ketika mengikuti referensi. Untuk lebih jelasnya, karakteristik respon saat diberi referensi *tracking* dapat dilihat pada Tabel 4.13.

Tabel 4.13 Karakteristik Respon Sistem pada Pembebanan Maksimal dengan Referensi *Tracking*.

Parameter	Nilai
Steady-State Error (%)	0,072
Presentase *Overshoot* (%)	0,013
RMSE	0,094



Gambar 4.12 Respon Sistem pada Pembebanan Maksimal dengan Referensi *Tracking*.

4.8 Implementasi Sistem

Setelah merancang dan menganalisa performa kontroler MPC pada tahapan simulasi, langkah selanjutnya adalah menerapkan dan mengimplementasikan kontroler pada sistem. Sebelumnya, akan dijelaskan telebih dahulu mengenai realisasi dari perancangan sistem yang telah dibuat pada Bab 3. Tahapan implementasi memiliki prosedur yang mirip dengan simulasi sistem. Hal yang berbeda terletak pada pergantian model matematika pada tahapan simulasi yang diganti dengan *plant* sesungguhnya melalui komunikasi *serial*. Untuk lebih jelasnya, blok diagram dan performa dari kontroler MPC pada tahapan sistem dapat dilihat pada subbab dibawah ini.

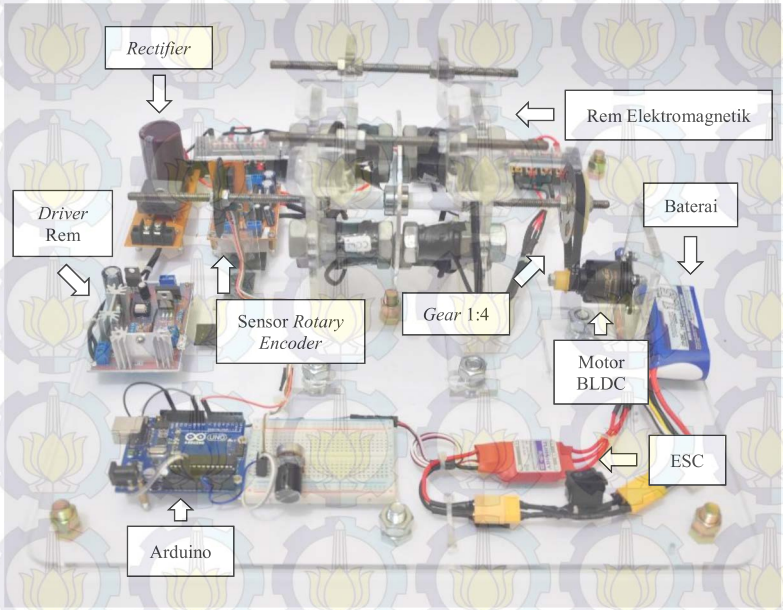
4.8.1 Realisasi Perancangan Sistem

Pada Bab 3, telah dijelaskan mengenai perancangan sistem untuk melaksanakan pengerjaan Tugas Akhir kali ini. Pada subbab ini, akan dijelaskan mengenai realisasi perancangan sistem keseluruhan, terutama bagian-bagian yang dikerjakan secara mandiri, seperti komponen sensor *rotary encoder* dan rem elektromagnetik. Selain itu, akan dijelaskan pula mengenai realisasi dan integrasi tiap-tiap komponen dari perancangan sistem secara keseluruhan.

Hal pertama yang akan dijelaskan terlebih dahulu adalah mengenai konstruksi sistem secara keseluruhan yang dapat dilihat pada Gambar

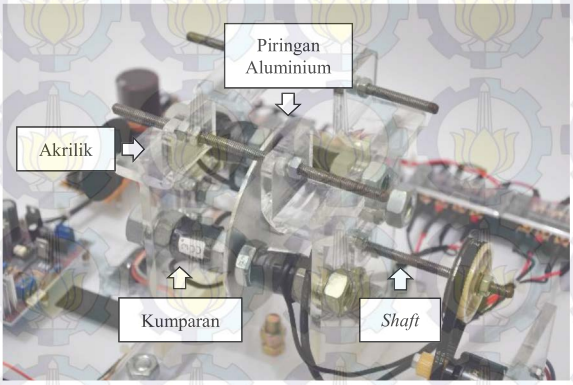
4.13. Pada gambar tersebut, terlihat bahwa motor BLDC dikopel dengan sistem *gear* 4:1 yang menghubungkan motor dengan *shaft*. Selain itu, motor BLDC tersebut disuplai tenaga dari baterai Lithium-Polimer dan dihubungkan ke ESC sebagai *driver* dari motor BLDC.

Sebagai salah satu komponen yang memiliki peranan yang sangat penting dalam memberi pembebanan pada motor, rem elektromagnetik dirancang untuk memberikan medan magnet yang kuat dengan suplai tegangan hingga 24 Volt. Rem tersebut tersusun dari kumparan lilitan tembaga yang mempunyai inti dari besi. Inti tersebut terbuat dari baut yang berdiameter 1,4 cm. Kumparan tersebut lalu disambung seri untuk tiap kutubnya dan dialiri tegangan yang diatur menggunakan metode PWM. Pengaturan tegangan yang masuk ke dalam kumparan ini diatur oleh driver dan Arduino. Sedangkan sumber tegangannya berasal dari listrik jala-jala PLN yang disearahkan dengan menggunakan komponen rectifier. Konstruksi rem elektromagnetik pada *plant* ditunjukkan seperti terlihat pada Gambar 4.14.

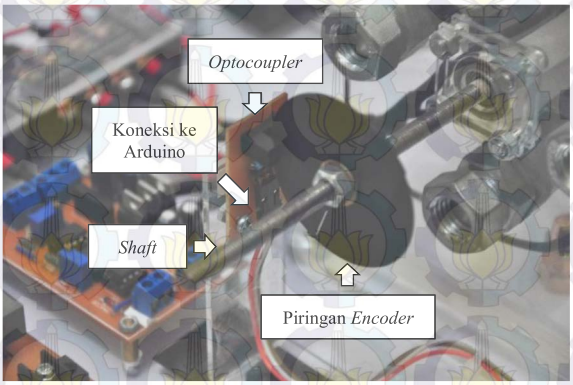


Gambar 4.13 Realisasi Perancangan Sistem Pengaturan Kecepatan Motor BLDC.

Selain rem elektromagnetik, komponen sensor *rotary encoder* juga mempunyai peranan yang penting untuk menghitung kecepatan dari motor BLDC. Sensor ini terdiri dari *optocoupler* dan resistor, serta diberi suplai tegangan sebesar 5 Volt. *Optocoupler* ini lalu dihubungkan dengan Arduino untuk menghitung banyaknya pulsa dalam satu detik, yang kemudian akan dikonversi ke dalam satuan kecepatan dengan algoritma tertentu. Hasil perancangan darisensor *rotary encoder* dapat dilihat pada Gambar 4.15.



Gambar 4.14 Konstruksi Rem Elektromagnetik pada Sistem.

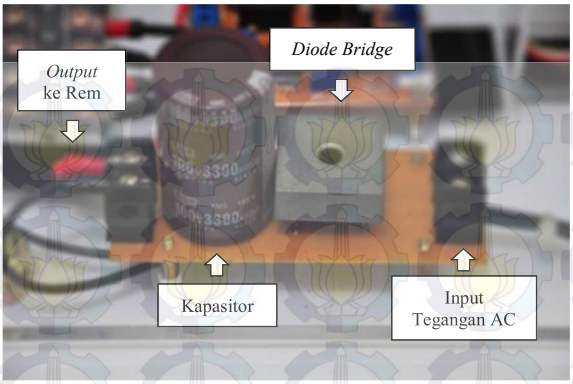


Gambar 4.15 Hasil Perancangan Sensor *Rotary Encoder*.

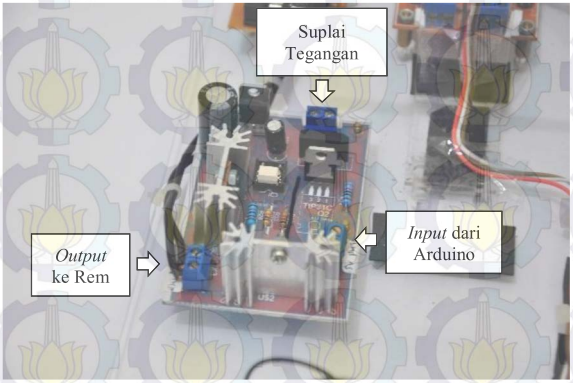
Mikrokontroler Arduino sebagai alat akuisisi data pada sistem ini juga mempunyai peranan yang sangat penting. Arduino merupakan tempat pengolahan dan algoritma penghitungan nilai kecepatan motor BLDC dilakukan. Selain itu, pada Arduino juga disambungkan sebuah *variable resistor*. *Variable resistor* (Varistor) ini berfungsi untuk mengubah masukan tegangan masukan pada rem elektromagnetik dari skala 0 sampai dengan 1000 melalui pin A0. Pada sisi lain, Arduino juga bertugas untuk memberi *output* PWM kepada ESC dan *driver* rem melalui pin 9 dan 11. Terakhir, *output* dari sensor *rotary encoder* disambungkan ke Arduino melalui pin 7. Tampilan mikrokontroler Arduino sebagai alat akuisisi data dapat dilihat pada Gambar 4.16.

Seperti telah disebutkan di atas, tegangan yang masuk ke dalam rem elektromagnetik diatur oleh sebuah *driver* dan rangkaian *rectifier*. Pada rangkaian *rectifier*, sumber yang digunakan adalah jala-jala listrik PLN dan keluaran yang dihasilkan berupa tegangan searah yang disambungkan ke dalam rem elektromagnetik. Hasil dari rangkaian *rectifier* dapat dilihat pada Gambar 4.17. Rangkaian terakhir adalah rangkaian *driver* rem yang berfungsi mengatur tegangan masukan ke rem dalam bentuk PWM. Rangkaian ini diberi input tegangan DC sebesar 12 Volt. Sedangkan input untuk mengatur besarnya nilai PWM berasal dari mikrokontroler Arduino. Rangkaian *driver* rem dapat dilihat pada Gambar 4.18.

Gambar 4.16 Alat Akuisisi Data berupa Mikrokontroler Arduino.



Gambar 4.17 Rangkaian *Rectifier* pada Sistem



Gambar 4.18 Rangkaian *Driver* untuk Pengaturan Tegangan *Input* pada Rem Elektromagnetik.

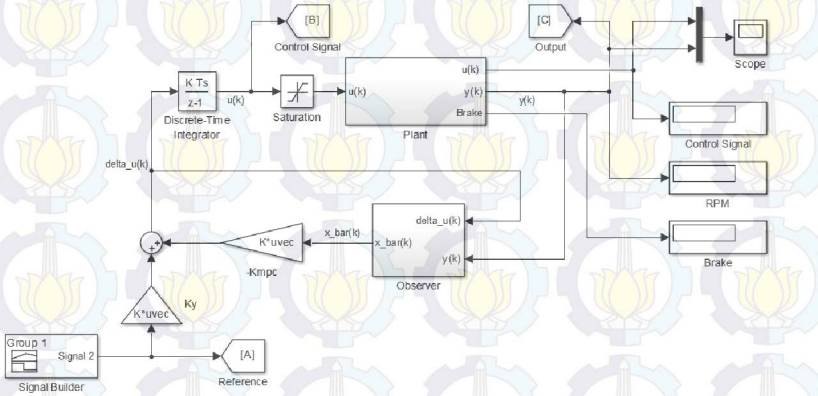
4.8.2 Diagram Blok Implementasi Sistem

Secara garis besar, perbedaan antara tahapan simulasi sistem dengan implementasi sistem hanya terletak pada pemodelan *plant*, khususnya motor BLDC. Jika pada tahap simulasi pemodelan *plant* digantikan dengan model matematika berbentuk *state-space*, tahap implementasi ini *output* kontroler langsung disalurkan ke dalam *driver* ESC pada *plant*. Untuk pembacaan nilai kecepatan motor BLDC dilakukan oleh sensor *rotary encoder* yang datanya dikirimkan kembali ke MATLAB melalui

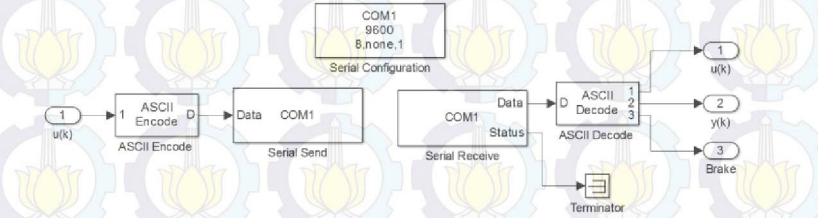
komunikasi *serial*. Selebihnya bentuk blok diagram implementasi sistem memilki kemiripan dengan tahapan simulasi sistem.

Pada Gambar 4.19, terlihat bahwa pemodelan matematika sistem dalam bentuk *state-space* telah digantikan dengan suatu subsistem bernama *plant*. Subsistem ini mempunyai 1 *input* berupa $u(k)$ atau sinyal kontrol dari kontroler MPC. Sedangkan *output*-nya berjumlah 3 yang terdiri dari sinyal kontrol $u(k)$, pembacaan nilai kecepatan dalam satuan RPM beserta nilai *input* pengereman (*brake*) yang diberikan pada sistem.

Isi dari subsistem *plant* yang merupakan komunikasi *serial* dapat dilihat pada Gambar 4.20 terlihat dari gambar bahwa blok komunikasi serial yang digunakan adalah *Serial Send*, *Serial Receive*, dan *Serial Configuration* yang terdapat pada *Instrument Control Toolbox*.



Gambar 4.19 Diagram Blok Implementasi Kontroler MPC pada Sistem.



Gambar 4.20 Subsistem *Plant* yang Terdiri Dari Komunikasi *Serial* pada Sistem.

Selain itu, digunakan blok *ASCII Decode* dan *ASCII Encode* yang berfungsi mengubah data menjadi menjadi bentuk ASCII yang dapat dibaca oleh Arduino.

4.8.3 Analisa Kontroler MPC pada Implementasi Sistem

Setelah menyiapkan diagram blok untuk tahap implementasi, langkah selanjutnya adalah menentukan nilai parameter kontroler MPC untuk tahapan implementasi sistem. Nilai parameter tersebut merupakan nilai *prediction horizon, control horizon* dan *tuning parameter* pada sistem. Nilai-nilai dari parameter tersebut didapat dari tahapan simulasi yang sebelumnya telah dilalui.

Berdasarkan tahapan simulasi, parameter dari kontroler MPC untuk tiap pembebanan dapat dilihat pada Tabel 4.14 berikut ini.

Setelah mengetahui nilai parameter kontroler MPC pada tiap-tiap pembebanan, langkah selanjutnya adalah menjalankan sistem yang sudah terpasang kontroler MPC dan melihat respon kecepatan yang diberikan pada tiap-tiap pembebanan dengan referensi *tracking*.

Terlihat semua pembebanan, pada respon yang diberikan oleh sistem dapat mengikuti referensi *setpoint* yang diberikan walaupun sistem, dalam hal ini motor BLDC, sedang diberikan pembebanan. Sistem dapat mengikuti referensi dengan baik pada kecepatan 1500 dan 2000 RPM. Akan tetapi, jika lebih diteliti secara seksama, terdapat *error steady-state* saat referensi berada pada kecepatan 2500 RPM. Respon yang diberikan juga terlihat tidak begitu halus mengikuti referensi *setpoint* yang diberikan.

Ada beberapa faktor yang menyebabkan respon terlihat tidak stabil di satu titik alias terdapat "riak" pada respon yang diberikan sistem. Faktor pertama berada pada sensor *rotary encoder* yang menerima banyak

Tabel 4.14 Nilai Parameter dari Kontroler MPC untuk Tiap–Tiap Nilai Pembebanan.

Parameter	Minimal	Nominal	Maksimal
Prediction Horizon N_p	50	40	60
Control Horizon N_c	2	3	4
Tuning Parameter r_w	0,1	0,2	0,3

noise dari lingkungan sekitar, contohnya adalah cahaya luar yang ikut masuk ke *optocoupler*, gangguan dari rem elektromagnetik yang disebabkan oleh kumparan rem yang jaraknya berdekatan dengan sensor, serta akurasi dari pembacaan dan kalibrasi pada sensor. Dikarenakan berbagai faktor diatas, pembacaan nilai kecepatan oleh sensor *rotary encoder* tidak bisa stabil di satu titik, melainkan naik dan turun.

Faktor kedua adalah pengestimasian nilai *state* lain yang tidak diketahui yang dilakukan oleh *observer* tidak dapat ditentukan secara akurat. Faktor ini disebabkan karena pembacaan sensor yang memang tidak stabil karena faktor pertama. Oleh sebab itu, *observer* tidak dapat mengestimasi nilai *state* lainnya secara akurat. Faktor terakhir adalah *data loss* yang terjadi pada komunikasi *serial*. *Data loss* terjadi apabila data *input* PWM yang dikirimkan MATLAB ke Arduino tiba-tiba terputus sehingga menganggu nilai *input* yang masuk ke *driver* ESC. Akibatnya, data *input* PWM yang dikirim ke *driver* ESC hilang untuk sepersekian detik sehingga kecepatan dari motor BLDC akan turun untuk sesaat. *Data loss* ini terjadi akibat *cache* dari mikrokontroler Arduino yang penuh, program *background* yang berjalan pada sistem operasi komputer yang dapat memperberat kinerja dari sistem dan faktor lainnya.

Subbab di bawah ini akan menjelaskan karakteristik respon untuk tiap pembebanan pada tahapan implementasi sistem.

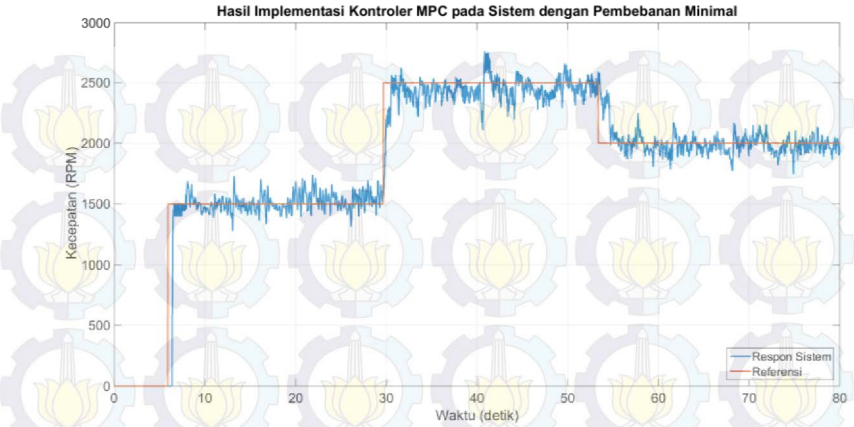
4.8.3.1 Beban Minimal

Pada pembebanan nominal, tegangan yang masuk ke dalam rem elektromagnetik sebesar 4,4 Volt untuk mengetahui apakah sistem dapat mengikuti referensi jika diberi beban berupa rem elektromagnetik. Sedangkan parameter kontroler MPC yang diberikan pada pembebanan nominal dapat dilihat pada Tabel 4.15.

Setelah tahap implementasi sistem pada pembebanan nominal, respon sistem lalu diamati untuk melihat karakterstiknya. Karakteristik respon untuk pembebanan minimal dapat dilihat pada Tabel 4.12.

Tabel 4.15 Karakteristik Respon dengan Pembebanan Minimal pada Implementasi Sistem.

Parameter	Nilai
Steady-State Error (%)	10
Rise Time (Detik)	0,78
Settling Time (Detik)	1,08
Presentase *Overshoot (%)*	0



Gambar 4.21 Hasil Implementasi Kontroler MPC pada Sistem dengan Pembebanan Minimal.

Pada Tabel 4.15, terlihat bahwa respon yang dihasilkan pada implementasi sistem mempunyai respon yang lebih cepat dibandingkan pada tahap simulasi. Akan tetapi, jika pada tahap simulasi tidak terdaapt *steady-state error* pada respon, hal ini berbeda dimana terdapat *steady-state error* sebesar 10%. Untuk tahap implementasi ini, tidak terdapat *overshoot* pada sistem, sama halnya seperti pada tahap simulasi. Respon *tracking* dan sinyal kontrol pada pembebanan minimal dapat dilihat pada Gambar 4.21.

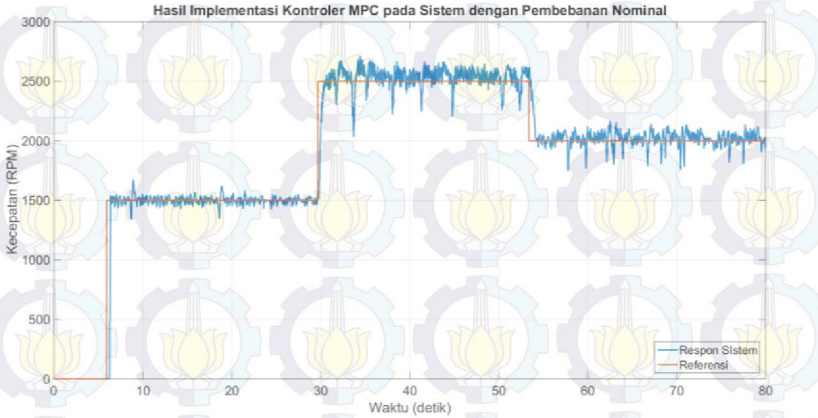
4.8.3.2 Beban Nominal

Setelah mengetahui respon sistem untuk pembebanan minimal, tahapan selanjutnya adalah mengetahui respon hasil implementasi untuk pembebanan nominal. Pada pembebanan nominal, sistem diberi beban berupa rem elektromagnetik dengan masukan tegangan sebesar 7,89 Volt. Sistem diberi referensi *tracking* dari kecepatan 1500 hingga 2000 RPM.

Setelah respon hasil implementasi didapat, langkah selanjutnya adalah mengamati karakteristik dari respon sistem saat pembebanan nominal. Karakteristik respon yang diamati adalah *settling time, rise time, error steady-state* dan *overshoot*. Karakteristik respon untuk pembebanan nominal dapat dilihat pada Tabel 4.16. Respon kecepatan dari motor

BLDC yang dihasilkan pada pembebanan nominal dapat dilihat pada Gambar 4.22.

Terlihat pada Tabel 4.16 bahwa masih terdapat nilai *steady-state error* pada pembebanan nominal, sama halnya seperti pada pembebanan minimal. Akan tetapi, nilai *steady-state error* pada pembebanan nominal mempunyai nilai lebih rendah dibandingkan pembebanan minimal. Untuk nilai *rise time* dan *settling time*, tahap implementasi mempunyai nilai yang lebih cepat dibandingkan tahap simulasi pada *software* MATLAB. Respon dari sistem lebih menyerupai orde 1 dikarenakan adanya nilai *overshoot*.



Gambar 4.22 Hasil Implementasi Kontroler MPC pada Sistem dengan Pembebanan Nominal.

Tabel 4.16. Karakteristik Respon dengan Pembebanan Nominal pada Implementasi Sistem.

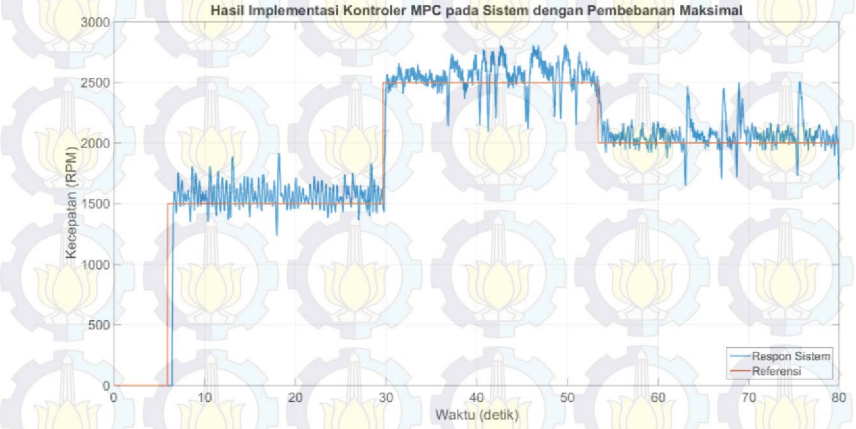
Parameter	Nilai
Steady-State Error (%)	7,4
Rise Time (Detik)	0,69
Settling Time (Detik)	1,27
Presentase *Overshoot (%)*	0

4.8.3.3 Beban Maksimal

Pada tahap implementasi terakhir, yaitu pada pembebanan maksimal, tegangan yang masuk ke dalam rem elektromagnetik sebesar 12,46 Volt untuk mengetahui apakah sistem dapat mengikuti referensi jika diberi beban berupa rem elektromagnetik. Referensi yang diberikan berupa *tracking* pada *setpoint* dengan nilai berkisar 1500 hingga 2500 RPM.

Setelah menyelesaikan tahap implementasi pada pembebanan maksimal, langkah selanjutnya adalah mengamati karakteristik respon dari sistem. Karakteristik respon yang diamati adalah settling *time, rise time, error steady-state* dan *overshoot*. Karakteristik respon untuk pembebanan maksimal dapat dilihat pada Tabel 4.17. Respon kecepatan dari motor BLDC yang dihasilkan pada pembebanan nominal dapat dilihat pada Gambar 4.23.

Tabel 4.17 menunjukkan bahwa masih terdapat *steady-state error* pada sistem, sebesar 8,9%. Nilai ini berbeda dibandingkan dengan tahapan simulasi dimana tidak adanya nilai *steady-stateg error* sama sekali. Sedangkan untuk kecepatan respon sistem, respon pada tahapan implementasi jauh lebih cepat dibandingkan pada simulasi. Ini dibuktikan dengan lebih kecilnya nilai *rise time* dan *settling time* pada tahap implementasi dibandingkan tahap simulasi.



Gambar 4.23 Hasil Implementasi Kontroler MPC pada Sistem dengan Pembebanan Maksimal.

Tabel 4.17 Karakteristik Respon dengan Pembebanan Maksimal pada Implementasi Sistem

Parameter	Nilai
Steady-State Error (%)	12
Rise Time (Detik)	0,94
Settling Time (Detik)	0.98
Presentase *Overshoot (%)*	0

DAFTAR PUSTAKA

[1] Xia, C.L., "*Permanent Magnet Brushless DC Motor Drives and Controls"*, John Wiley & Sons Singapore Pte. Ltd., Singapore, 2012.

[1] Elrosa, Ilmiyah., "Traction Control pada Parallel Hybrid Electric Vehicle dengan Metode Generalized Predictive Control", *Tugas Akhir*, Jurusan Teknik Elektro, Institut Teknologi Sepuluh Nopember, Surabaya, 2014

[2] Janardana, Gede B.A., "Desain dan Analisis Motor Axial Flux Brushless DC Berbasis 3D Finite Element Method Untuk Aplikasi Kendaraan Listrik", *Tugas Akhir*, Jurusan Teknik Elektro, Institut Teknologi Sepuluh Nopember, Surabaya, 2015

[3] _____, "*2-2-1 What is a Brushless DC Motor ?*", <URL: http://www.nidec.com/en-NA/technology/motor/basic/00018/>, Maret, 2015

[4] _____, "*Brushless DC Motor*", <URL: http://hades.mech.northwestern.edu/index.php/Brushless_DC_Motors>, Maret, 2014

[5] _____, "*How Hall-Effect Sensors Impact Motor Energy Use*", <URL: http://www.automationworld.com/sensors-discrete/how-hall-effect-sensors-impact-motor-energy-use>, Maret, 2015

[6] _____, "*Brushless Motor DC Animation*", <URL: http://www.avdweb.nl/solar-bike/hub-motor/permanent-magnet-dc-hub-motor-tuning.html>, Maret, 2015

[7] _____, "*How are Magnets Used in Amusement Park Rides?*", <URL: https://period7magnets.wikispaces.com/How+are+magnets+used+in+amusement+park+rides%3F>, April, 2015

[8] Permana, Yoki., "Perancangan dan Implementasi Pengaturan Kecepatan Motor 3 fasa pada Mesin Sentrifugal Menggunakan Metode Model Reference Adaptive Control (MRAC)", *Tugas Akhir*, Jurusan Teknik Elektro, Institut Teknologi Sepuluh Nopember, Surabaya, 2014

[9] _____, "*The MCU and Other Components of The NC System*", <URL: https://rekadayaupaya.wordpress.com/2013/06/13/8-4-the-mcu-and-other-components-of-the-nc-system/>, April, 2015

[10] Söderström, T. and Stoica, P., "*System Identification"*, Prentice Hall, United Kingdom, 2001

[11] Wang, L., "*Model Predicitive Control System Design and Implementation Using MATLAB*", Springer, London, 2009

[12] Seborg, D.E., Mellichamp, D.A., Edgar, T.F., dan Doyle, F.J., "*Process Dynamic and Control*", Wiley, London, 2011
[13] _____, "*HP2212-1000KV Outrunner Brushless Motor For Quadrotor*", <URL: http://rctimer.com/product-813.html>, April, 2015
[14] _____, "*TURNIGY Plush 30Amp Speed Controller*", <URL: http://www.hobbyking.com/hobbyking/store/__2164__TURNIGY_Plush_30amp_Speed_Controller.html>, April, 2015
[15] _____, "*SparkFun Low Current Sensor Breakout - ACS712*", <URL: https://www.sparkfun.com/products/8883>, April, 2015
[16] _____, "*Arduino*", <URL:http://en.wikipedia.org/wiki/Arduino>, April, 2015
[17] Ogata, Katsuhiko., "*Discrete-Time Control System*", Prentice Hall, New Jersey, 1994
[18] Ogata, Katsuhiko., "*Modern Control Engineering 4th Edition"*, Prentice Hall, New Jersey, 2002

BAB 5
PENUTUP

Bab 5 merupakan bab terakhir pada laporan Tugas Akhir ini yang akan menjelaskan tentang kesimpulan yang dapat ditarik setelah melakukan proses pengerjaan Tugas Akhir ini. Pada bab ini juga disertakan mengenai saran yang dapat dipertimbangkan dalam pengerjaan Tugas Akhir selanjutnya.

5.1 Kesimpulan

Berdasarkan hasil simulasi dan implementasi sistem pengaturan kecepatan motor BLDC menggunakan kontroler *Model Predictive Control* (MPC), dapat ditarik beberapa kesimpulan sebagai berikut:

a. Hasil simulasi dan implementasi kontroler MPC pada sistem pengaturan kecepatan motor BLDC menunjukkan bahwa kecepatan motor BLDC dapat mengikuti referensi *tracking* pada semua parameter pembebanan (minimal, nominal dan maksimal).
b. Penentuan nilai *prediction horizon* pada kontroler MPC sangat mempengaruhi respon dari sistem. Semakin besar nilai *prediction horizon*, respon yang dihasilkan akan semakin lambat dan halus. Sebaliknya, semakin kecil nilai *prediciton horizon* akan mengakibatkan respon yang semakin cepat. Akan tetapi, nilai *prediciton horizon* yang semakin kecil dapat menimbulkan *overshoot* yang semakin tinggi.
c. Selain *prediction horizon*, parameter *control horizon* pada kontroler MPC juga memilki pengaruh yang besar pada sistem. Jika nilai *control horizon* semakin besar, maka sistem akan memiliki waktu respon yang lebih cepat dengan kekurangan *overshoot* yang semakin tinggi. Adapun jika nilai *control horizon* semakin kecil, respon yang dihasilkan sistem akan semakin halus walaupun waktu respon akan semakin lambat.
d. Pada parameter terakhir, yaitu *tuning parameter* pada indeks performansi, juga mempunyai pengaruh yang signifikan pada sistem. Semakin besar nilai *tuning parameter*, respon yang dihasilkan akan semakin cepat walaupun terdapat kekurangan karena timbulnya *overshoot*. Sebaliknya, semakin kecil nilai

tuning parameter, respon yang dihasilkan akan semakin lambat dan menyerupai respon orde 1 tanpa adanya *overshoot*.

5.2 Saran

Setelah melalui banyak proses dan tahapan pada pengerjaan Tugas Akhir, penulis dapat memberikan berapa saran seperti berikut ini:

a. Terdapat beberapa parameter dari lingkungan luar yang dapat mempengaruhi kecepatan dari motor BLDC pada sistem atau *plant* ini. Antara lain kondisi permukaan pada *plant*, posisi *belt* yang menghubungkan motor dengan *shaft* dan beberapa parameter lainnya. Oleh karena itu, untuk pengerjaan Tugas Akhir selanjutnya dapat memperhatikan dan memperbaiki hal-hal minor diatas.
b. Pada pengerjaan Tugas Akhir ini, sumber tenaga yang digunakan untuk menjalankan motor BLDC adalah baterai *Lithium-Polimer* (Li-Po). Disarankan untuk pengerjaan Tugas Akhir selanjutnya, lebih mengutamakan penggunaan sumber tenaga listrik yang permanen (contohnya listrik PLN). Ini dikarenakan kapasitas dari baterai yang dapat mempengaruhi kecepatan dari motor BLDC.
c. Pada pengerjaan Tugas Akhir selanjutnya disarankan untuk mengganti sensor *rotary encoder* dengan menggunakan sensor *tachogenerator* ataupun sensor *rotary encoder* dengan standar industri. Ini untuk menjamin bahwa pembacaan kecepatan motor BLDC dapat berjalan secara presisi dan akurat.
d. Pada perancangan kontroler *Model Predictive Control* (MPC), disarankan untuk mengaplikasikan kontroler MPC ini dengan batasan atau *constraint*, baik constraint pada sinyal kontrol ataupun sinyal *output*. Pengaplikasi *constraint* pada kontroler MPC dapat mencegah *overshoot* dan sinyal kontrol berlebihan yang masuk ke dalam *driver* motor BLDC sehingga dapat mengurangi resiko rusaknya *driver* motor BLDC.

LAMPIRAN

A. Program Akuisisi Data Arduino.

```
#include <Servo.h>
Servo esc;
int setpoint;
int incomingByte;
int percenttrot;
int encoderPin = 7;
unsigned long duration;
unsigned long rpm;
char disp2[1];
char disp3[1];

void setup()
{
  Serial.begin(115200);
  pinMode(encoderPin,INPUT);
  esc.attach(9);
  esc.writeMicroseconds(1080);
  delay(2000);
}

void loop()
{
  setpoint = map(percenttrot,0,1000,1080,1750);
  esc.writeMicroseconds(setpoint);
  char disp1[10];
  if((setpoint<=1751)&&(setpoint>=1079))
  {
    sprintf(disp1,"*%4d",percenttrot);
    Serial.print(disp1);
  }
  else
  {
    setpoint=1080;
    percenttrot=0;
    sprintf(disp1,"*%4d",percenttrot);
```

```
    Serial.print(disp1);
  }
  if (Serial.available()>0)
  {
    percenttrot = Serial.parseInt();
  }
  duration = pulseIn(encoderPin,HIGH,100000);
  ram =(79550000/3)/duration;
  if (rpm>4000)
  {
    rpm = 0;
  }
  Serial.print('#');
  sprintf(disp2,"%4d",rpm);
  Serial.print(disp2);
  int sensorValue = analogRead(A0);
  int rem = map(sensorValue,0,1023,1,254);
  analogWrite(11,rem);
  int percentrem = map(sensorValue,0,1023,1,999);
  int arus = analogRead(A2)*4;
  sprintf(disp3,"%3d$%3d",percentrem,arus);
  Serial.print('@');
  Serial.print(disp3);
}
```

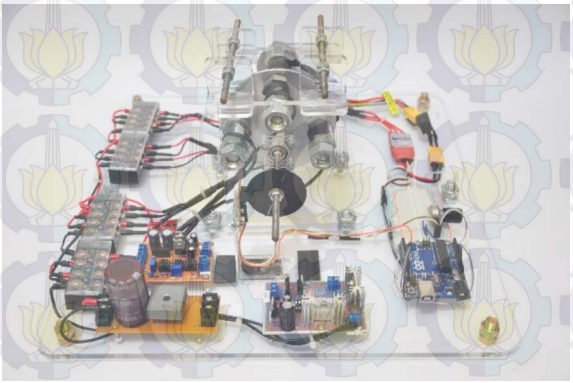
B. Program Pencarian Nilai *Root Mean Square Error* (RMSE).

```
clc;
n=size(output_40,1);
real=output_40(:,1);
model=output_40_model_arx3(:,1);
sum_err=0;
for i=1:n;
    if real(i)==0;
        real(i)=0.01;
        model(i)=0.01;
    end
    err=(real(i)-model(i))/real(i);
    sum_err=sum_err+err^2;
end
sum_err;
```

```
n;
rmse_val=sqrt(sum_err/n)
```

C. Hasil Perancangan Sistem Pengaturan Motor BLDC.

Perancangan sistem ini dikerjakan oleh BLDC Team yang beranggotakan 5 orang. Kelima orang tersebut adalah Fachrul Arifin, Beny Setyiadi Hidayat, Hudaibiy Hibban, Habib Ibnu Hasan dan Muhammad Ammar Huwaidi. Sistem dan *plant* ini kami beri nama BLDC V-1. Sistem ini dikerjakan selama satu semester dan dapat dijadikan acuan dan referensi sistem pada pengerjaan Tugas Akhir yang akan datang.



Gambar 1. *Plant* Sistem Pengaturan Kecepatan Motor BLDC dengan nama BLDC V-1

D. Program MATLAB Kontroler *Model Predictive Controller*.

```
clc;
clear;

%State space sistem
Ap=[0 1;0.005205 0.9936];
Bp=[0;1];
Cp=[0.006564 0.002823];
Dp=0;

%Parameter kontroler
```

```
Np=500; %prediction horizon
Nc=13; %control horizon
r_w=10; %tuning paramter

%Augmented model
[m1,n1]=size(Cp);
[n1,n_in]=size(Bp);
A_e=eye(n1+m1,n1+m1);
A_e(1:n1,1:n1)=Ap;
A_e(n1+1:n1+m1,1:n1)=Cp*Ap;
B_e=zeros(n1+m1,n_in);
B_e(1:n1,:)=Bp;
B_e(n1+1:n1+m1,:)=Cp*Bp;
C_e=zeros(m1,n1+m1);
C_e(:,n1+1:n1+m1)=eye(m1,m1);

%Matriks Phi dan F
n=n1+m1;
h(1,:)=C_e;
F(1,:)=C_e*A_e;
for kk=2:Np
    h(kk,:)=h(kk-1,:)*A_e;
    F(kk,:)= F(kk-1,:)*A_e;
end
v=h*B_e;
Phi=zeros(Np,Nc);
Phi(:,1)=v;
for i=2:Nc
    Phi(:,i)=[zeros(i-1,1);v(1:Np-i+1,1)];
end
BarRs=ones(Np,1);
Phi_Phi= Phi'*Phi; %Matriks Phi_Phi
Phi_F= Phi'*F; %Matriks Phi_F
Phi_R=Phi'*BarRs; %Matriks Phi_R
Q=zeros(1,Nc);
Q(1,1)=1;

%Mencari gain Kx,Ky
Kmpc=Q*inv(Phi_Phi+(r_w*eye(Nc,Nc)))*(Phi_F);
```

```
Ky=Q*inv(Phi_Phi+(r_w*eye(Nc,Nc)))*(Phi_R);
Kx=Kmpc(:,1:2);

%Mencari gain observer
pole=[-0.001 -0.12 -0.03];
Kob=place(A_e',C_e',pole)';
```